Constant-size Lattice-Based Group Signature with Forward Security in the Standard Model *

Sébastien Canard[1], Adela Georgescu[2,3], Guillaume Kaim[1,2], Adeline Roux-Langlois[2] and Jacques Traoré[1]

[1] Orange Labs, Applied Crypto Group, Caen, France
[2] Univ Rennes 1, CNRS, IRISA
[3] Department of Computer Science, University of Bucharest

Abstract. One important property of group signatures is forward-security, which prevents an attacker in possession of a group signing key to forge signatures produced in the past. In case of exposure of one group member's signing key, group signatures lacking forward-security need to invalidate all group public and secret keys (by re-initializing the whole system) but also invalidate all previously issued group signatures. Most of the existing forward-secure group signatures (FS-GS) are built from number-theoretic security assumptions which are vulnerable to quantum computers. The only post-quantum secure FS-GS scheme is built from lattices by Ling et al. (PQCrypto 19) in the random oracle model, following the classical framework of encrypt-then-prove, thus using non-interactive zero-knowledge (NIZK) proofs. In this work, we achieve the first FS-GS from lattices in the standard model. Our starting point is the group signature of Katsumada and Yamada (Eurocrypt 19) which replaces NIZK by attribute-based signatures (ABS), thus removing the need for random oracles. We first modify the underlying ABS of Tsabary (TCC 17) to equip it with forward-security property. We then prove that by plugging it back in the group signature framework of Katsumada and Yamada (Eurocrypt 19), we can design a FS-GS scheme secure in the standard model with public key and signature size constant in the number of users. Our constant size is achieved by relying on complexity leveraging, which further implies relying on the subexponential hardness of the Short Integers Solution (SIS) assumption.

1 Introduction

Group signatures were introduced as a new type of signatures by Chaum and van Heyst [CvH91] in 1991 and they were designed to allow only members of a group to sign messages while the identity of the signer remains hidden for the verifier (anonymity). The latter can only ensure that a member belonging to the group has signed the message. Moreover this property guarantees the unlinkability as

* This paper is an extended version of the article published in the Proceedings of ProvSec 2020, LNCS 12505, Springer-Verlag, which is available online at `https://doi.org/10.1007/978-3-030-62576-4_2`

well, preventing anyone to detect that two group signatures have been generated by the same group member. Nevertheless, if necessary, the signature can be opened by an entity called group manager who holds some secret information and reveals the identity of the signer (traceability). These features make group signatures very useful for real life applications including e-commerce systems, anonymous online communications and trusted hardware attestations.

From their beginning until now, a great variety of constructions for group signatures have been proposed, addressing different needs: in the random oracle model [CL04,BBS04] or standard model [BW06,Gro07], supporting static groups [BMW03], dynamic groups [BSZ05] or partially dynamic groups and constructions based on different theoretical assumptions such as RSA [ACJT00], or pairings [BBS04] for standard assumptions.

As for post-quantum constructions, there is a vast literature concerning group signatures based on lattices, some of them being designed to support most of the important properties listed above. Among them there are group signatures in the static model [GKV10,CNR12,LLLS13,NZZ15,LNW15,LLNW16], [LNWX18,BCN18,dPLS18], in the dynamic model where users have the flexibility to join and leave the group [LNWX17], achieving partially dynamicity by means of verifier-local revokability [LLNW14] (where new users can not join the group but they can leave it being revoked), or using other tools to achieve partial dynamicity [LLM$^+$16,LMN16]. Concerning the random oracle model (ROM) and the standard model, all of the existing lattice-based group signature schemes are in the ROM except the construction by Katsumata and Yamada [KY19]. We can further notice that the recent construction of non-interactive zero-knowledge proof of knowledge (or NIZKPoK or NIZK) for all NP from [PS19] combined with [BMW03] can be adapted in a group signature scheme in the standard model in a straightforward manner (but very inefficiently as we explain in the next subsection).

Forward-security [Son01,NHF09,LY10] is an important additional security property sometimes considered in group signature constructions. Concerning lattices, to the best of our knowledge there is one such construction in the ROM model [LNWX19]. This property cuts the time into periods t and prevents attackers from forging group signatures pertaining to past time periods $t' < t$, even if a secret group signing key is revealed at the current time period. As explained in Song [Son01], in the context of group signatures, exposure of secret signing keys is more damaging since an adversary being in the possession of a member's group signing key can produce signatures on behalf of the whole group, but still remaining anonymous. As a consequence, all the public and secret keys of the group need to be regenerated and all previously generated group signatures have to be rendered invalid. We note that the solution to these problems is adding a forward-secure mechanism to group signatures as was previously done first for key exchange protocols ([Gün89],[DvOW92]) and then for digital signatures ([BM99],[IR01]), symmetric-key encryption ([BY03]) and public key encryption systems ([CHK03]). This property aims to protect past use of private keys even if an adversary breaks-in at the current moment of time. When entering a new

time period t, a new secret key related to t is computed from the previous secret key related to $t-1$ through a one-way key evolution algorithm, the latter one being deleted promptly afterwards.

1.1 Our contribution

Our main achievement in this paper is a lattice-based forward-secure group signature scheme without NIZK in the standard model with public key and signature size constant (independent) on the number of users. We note that this is the first of its kind in the standard model as the only existing construction [LNWX19] for group signatures from lattices achieving forward-security is in the random oracle model. Our group signature corresponds to the transformation of the lattice-based group signature from [KY19] using the idea from [LNWX19] to obtain forward-security.

The main building block of our transformation is a lattice-based forward-secure attribute-based signature scheme (FS-ABS) that we introduce later in this paper as a novelty. We mention that there is a previous general construction of FS-ABS due to [YLH+12] which combines the general primitive of credential bundles (which can be instantiated with forward-secure digital signatures) and non-interactive witness-indistinguishable (NIWI) proofs. Using this framework one can achieve lattice-based FS-ABS as long as one can build forward-secure digital signature and NIWI from lattices, but, to the best of our knowledge, there is no construction in the lattice-settings for any of them. Therefore, we believe that our construction of lattice-based FS-ABS is of self-interest.

We mention that our group signature scheme satisfies CCA-selfless-anonymity (inherited from the base group signature scheme of [KY19]), a relaxation of the CCA-full-anonymity, since the adversary is not in possession of all the secret keys: he is missing the secret keys of the two members, whose identities compose the challenge. As for the traceability property, we show that our scheme achieves forward-secure traceability.

As already explained in [KY19], group signatures from lattices in the standard model can be achieved also by using the recent proposal for NIZK for all NP from LWE [PS19] (published shortly after [KY19]) instead of the ABS. The difference is that the ABS that we employ in our group signature scheme relies only on the hardness of SIS, avoiding the potentially stronger LWE assumption on which the NIZK mentioned above relies, leading to a potentially heavier group signature construction. Another drawback of a potential instantiation of group signatures using [PS19] is that the latter one relies on fully homomorphic encryption for evaluating circuits making it very costly in time efficiency.

1.2 Overview of the building blocks for our construction

In order to have all the elements needed to give a technical overview of our scheme, we start by describing three existing constructions: the group signature scheme of [KY19], the ABS proposed by Tsabary [Tsa17], and finally, the forward-secure mechanism of the group signature construction of [LNWX19].

Group Signature Scheme without NIZK. The starting point of our work is the recent lattice-based group signature scheme without NIZK in the standard model [KY19]. Previous to this construction, all works on group signatures were relying on the Sign-Encrypt-Prove framework defined by Bellare, Micciancio and Warinschi [BMW03]. In this framework, to sign a message M, a user encrypts both his certificate received from the group manager and a digital signature on M. Finally, he proves in non-interactive zero-knowledge that every element is well formed. Until recently (2019), constructing NIZK from lattices for any NP language was a long-standing open problem and by that time Katsumata and Yamada [KY19] proposed a group signature scheme that by-passed the utilization of NIZK by replacing it with *indexed* attribute-based signature scheme (ABS). Their idea is based on the fact that for group signatures the needed NIZK is in the common reference string (CRS) model and, in the context of group signatures, it resembles to *designated-prover* NIZK (DP-NIZK) where there is a proving key k_P that needs to be kept secret (and thus is not known to the verifier, assuring zero-knowledge) and a verification key k_V which is public. Anyway, simply replacing NIZK in the CRS model with DP-NIZK is not enough since it trivially breaks anonymity. The breakthrough idea of Katsumata and Yamada was to view ABS as DP-NIZK. In attribute-based signatures, a signer with an attribute x is provided a secret key sk_x from the authority and can anonymously sign a message associated with a policy C using his secret key, if and only if, his attribute satisfies the policy C. In particular, the signature hides the attribute (anonymity) and users can not collude to pull their attributes together if none of the attributes satisfies the policy associated to the message (unforgeability). Now, an ABS can be seen as a DP-NIZK by the following association: the attribute x is seen as a witness w and the ABS signing key sk_x can be set as the proving key k_P of the DP-NIZK. Thus proving that w is a valid witness to a statement s i.e. $(s, w) \in \mathcal{R}$ for the NP relation $\mathcal{R}$ resorts to, firstly prepare a circuit $C_s(w) = \mathcal{R}(s, w)$ that has the statement s hard-wired into it, secondly sign a message associated with the policy C_s using the proving key $k_P = sk_x$ and finally output the signature as the NIZK proof π. Anonymity and unforgeability of the ABS assure the zero-knowledge property and soundness respectively.

Having shown a way of substituting the NIZK with ABS, it remains to indicate how to use ABS to construct group signature. We briefly explain, in the following, the general framework from [KY19]. The group manager issues for user i a key K_i of a secret key encryption (SKE) scheme and an ABS signing key $sk_{i||K_i}$ where $i||K_i$ is seen as an attribute. To sign a message M, the group member i encrypts his identity under K_i obtaining $\mathsf{ct}_i = \mathsf{SKE.Enc}(K_i, i)$ and creates an attribute-based signature for some policy C_{ct_i} which serves as a NIZK proof of the fact that ct_i encrypts the identity. The circuit C_{ct_i} has the statement ct_i hardwired such that $C_{\mathsf{ct}_i}(i||K_i) := (i = \mathsf{SKE.Dec}(K_i, \mathsf{ct}_i))$. The traceability property of the group signature holds from unforgeability of ABS and anonymity holds from anonymity of the ABS and semantic security of the SKE.

As for the instantiation of the ABS from lattices, [KY19] gives two possible solutions: the first one uses the ABS proposed by Tsabary [Tsa17] proven secure

under the SIS assumption and the second one is an indexed ABS designed by them, relying also on the SIS assumption. The need for the second construction is explained by the problems encountered when trying to plug the first construction into a group signature. Tsabary's scheme achieves selective unforgeability which is not enough for security purposes of group signatures. Adaptiveness is the required property and can be easily achieved via complexity leveraging with the drawback that this approach requires a subexponential security loss. We remark that in [KY19] they emphasize that they don't really need adaptiveness but rather something complementary to selectiveness called co-selective unforgeability. Unfortunately, we can not achieve this property directly, without complexity leveraging (see section 3.3 for more details). The two different ABS constructions give two different group signature schemes with the following properties:

(i) Tsabary's ABS gives rise to a group signature scheme with public key and signature size constant (independent) in the number of users and whose security relies on the hardness of LWE with polynomial approximation factor and subexponential hardness of SIS with polynomial approximation factor.
(ii) The second ABS gives rise to a group signature scheme with public key and signature size linear in the number of users whose security relies on the hardness of LWE and SIS with polynomial approximation factors.

Attributed Based Signature from Constrained Signature of Tsabary. The main building block of our group signature is an Attribute Based Signature scheme. In the following we briefly explain the ABS developped in Tsabary's paper. First of all, the construction in Tsabary's paper is not really an *attribute-based signature* but rather a *key-policy constrained signature* or simply *constrained signature*. We note that the other flavour of constrained signatures, as defined in [Tsa17], called *message-policy constrained signature* is equivalent to attribute-based signatures. In constrained signatures, a signing key sk_f is associated with a policy $f : \{0,1\}^* \to \{0,1\}$, called the constraint, and a key sk_f can sign a message $x \in \{0,1\}^*$ only if the message satisfies the policy i.e. $f(x) = 0$. In attribute-based signatures each key is associated with an attribute $x \in \{0,1\}^*$ and a key sk_x can sign a policy f only if the attribute satisfies the policy i.e. $f(x) = 0$. A constrained signature can be easily transformed into an attribute-based signature using universal circuits (which we denote U_x) as briefly explained in [KY19] (but not done there), transformation that we apply in Section 3 and that we sketch below.

The ABS scheme (as well as the original constrained signature of [Tsa17]) is built from lattice trapdoors. The verification key vk consists of a uniformly sampled matrix $\overrightarrow{\mathbf{A}} = [\mathbf{A}_1\|...\|\mathbf{A}_\ell] \in \mathbb{Z}_q^{n\times(m\times\ell)}$ (with ℓ the input size of the circuit C) and a close to uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times p}$ while the master signing key msk is a trapdoor for $\mathbf{A}$ denoted $\mathbf{A}_{\gamma_0}^{-1}$. The signing key sk_{x_i} is associated to an user i (we prefer the simplified version of this notation even though it would be clearer to use $sk_{U_{x_i}}$ as notation) and to an universal circuit U_{x_i} (which has the attribute hard-wired and takes as input the policy (circuit) and a message). The secret key sk_{x_i} is a trapdoor $[\mathbf{A}\|\mathbf{A}_{x_i}]_\gamma^{-1}$ where $\mathbf{A}_{x_i} = \overrightarrow{\mathbf{A}} \cdot \mathbf{H}_{U_{x_i}} \in \mathbb{Z}_q^{n\times m}$ is

computed from $\overrightarrow{\mathbf{A}}$ and U_{x_i} using the function EvalF. This function, associated with a function EvalFX, allows to compute $\mathbf{H}_{U_{x_i}} = \mathsf{EvalF}(U_{x_i}, \overrightarrow{\mathbf{A}})$, and $\mathbf{H}_{U_{x_i},\mathbf{x}} = \mathsf{EvalFX}(U_{x_i}, \mathbf{x}, \overrightarrow{\mathbf{A}})$ both in $\mathbb{Z}^{(\ell m)\times m}$ and of bounded norm such that $(\overrightarrow{\mathbf{A}} - \mathbf{x} \otimes \mathbf{G}) \cdot \mathbf{H}_{U_{x_i},\mathbf{x}} = \overrightarrow{\mathbf{A}} \cdot \mathbf{H}_{U_{x_i}} - U_{x_i}(\mathbf{x})\mathbf{G} \mod q$, where $\mathbf{G}$ is the gadget matrix. Then, the manager can easily generate the secret key sk_{x_i} using it's own trapdoor $\mathbf{A}_{\gamma_0}^{-1}$. A valid signature for a message M, a circuit C and an attribute x_i is a short vector σ such that $[\mathbf{A} \| \overrightarrow{\mathbf{A}} - x_i \otimes \mathbf{G}] \cdot \sigma = \mathbf{0} \mod q$. We note that for every tuple (C, M, x_i), a trapdoor $[\mathbf{A} \| \overrightarrow{\mathbf{A}} - x_i \otimes \mathbf{G}]_{\gamma'}^{-1}$ can be derived from $[\mathbf{A} \| \overrightarrow{\mathbf{A}} - U_{x_i}(C, M)\mathbf{G}]_{\gamma'}^{-1}$ when $U_{x_i}(C, M) = C(x_i) = 0$.

We remark that, at this stage, the unforgeability of the ABS can be easily broken, as explained in [KY19] because the message is not bounded to the signature (both signature and verification just ignore the message) and a valid signature for a pair of policy and message (C, M) is also valid for (C, M') for $M \neq M'$. Therefore, in the security game, we can not allow signature queries and following the idea of [KY19], we use the fact that a scheme that is unforgeable only when the adversary can not make signature queries can be generically transformed into a scheme that is unforgeable even when the adversary is allowed to make signature queries. In short, the idea in [KY19] is to answer the signing queries using the secret key of a dummy user which does not exist in the real system. We will need to partition the set of all possible message-policy pairs into a challenge set and a controlled set (using admissible hash functions) with the hope that the adversary asks queries that fall into the controlled set to which the challenger can answer with the help of the dummy key. We also hope that the attacker outputs a forgery in the challenge set to allow the simulator to solve a hard problem.

Forward Secure Group Signature of [LNWX19]. Recall that for achieving forward-secure group signature, one needs a one-way key evolving mechanism for deriving secret keys for every period of time. Let us now briefly explain this mechanism following the idea of [LNWX19]. Let $T = 2^d$ be the total number of time periods, the time periods are represented in a binary tree, where each time period is a leaf of the tree. Each user secret key for a time period t is then associated with a sub-tree of depth d which uniquely defines the time period t. Let z be a binary string (corresponding to a time period) of lenth d_z. The set $\text{Nodes}_{(t \leftarrow T-1)}$ contains nodes for which bases (trapdoors) are derived at a current period of time t and which allow to compute subsequent keys in the key update algorithm using the bonsai tree technique [CHKP10]. Each user will have associated a matrix corresponding to period time $z \in \text{Nodes}_{(t \leftarrow T-1)}$: $\mathbf{A}_{x_i,z} = [\mathbf{A} \| \mathbf{A}_{x_i} \| \mathbf{T}_1^{z[1]} \| \cdots \| \mathbf{T}_{d_z}^{z[d_z]}]$ where the last d_z matrices corresponding to the bits of d_z are public. Therefore, the group signing key of user i at time t is $\{\mathbf{S}_{i||z}, z \in \text{Nodes}_{(t \leftarrow T-1)}\}$ which satisfies $\mathbf{A}_{x_i,z} \cdot \mathbf{S}_{i||z} = \mathbf{0} \mod q$. The user is then able to compute all possible $\mathbf{S}_{i||t}$ by employing $\mathbf{S}_{i||z}$ if z is an ancestor of t where t is the binary representation of a period of time. The basis delegation technique

allows users to compute trapdoor matrices for all the descendent nodes in the set $\text{Nodes}_{(t \leftarrow T-1)}$ and therefore to compute all the subsequent signing keys.

1.3 Our construction

We are now able to better explain our contribution. We start from the constrained signature of Tsabary, we transform it in an ABS (according to [KY19] suggestion) as previously explained, we equip it with forward-security (following the mechanism of [LNWX19]), then plug it into the group signature of [KY19]. Thus we achieve the first forward-secure group signature from lattices without NIZK in the standard model having public key and signature size independent of the number of users for which we managed to prove forward-secure traceability and CCA-selfless anonymity. The drawback is that the security assumption on which the GS scheme relies is SIS with subexponential hardness.

Our main building block is then a forward secure Attribute Based Signature which is built using the idea from [LNWX19] having as starting point Tsabary's constrained signature. As explained in [LNWX19], the advantage of this method is that it incurs only logarithmic dependency on T. Therefore our construction achieves signature size and public key size constant in N and logarithmic in T. We note that [LNWX19] applied it directly for building forward-secure group signature (FS-GS) while we need to apply it first on our ABS to get forward-secure attribute-based signatures (FS-ABS). Indeed, in an encrypt-then-prove paradigm for group signatures, the transformation of [LNWX19] into a forward secure group signature is independent of the encryption scheme and of the NIZK scheme used to prove the membership. This is because the group secret key of a user does not appear as input in the NIZK proof but is embedded in a ciphertext on which the proof is performed. Instead, the paradigm on which we build our construction uses an ABS to prove that the user belongs to the group, and the ABS secret key is a direct component of the group secret key of a user. This means that if we want to update the group secret key of a user, we need to update the ABS secret key as well.

From this observation and the fact that the ABS built by Tsabary [Tsa17] is based on lattice trapdoors which fit perfectly with bonsai trees, we can then adapt the forward security mechanism of [LNWX19] to the ABS derived from [Tsa17], and use the resulting ABS to get forward-secure group signature scheme. We note that if we try to apply the same technique for the second ABS from [KY19] (also built from lattice trapdoors) we can not get forward-security. The problem is that the design of ABS forces us to keep the initial secret key derived by the master authority for every user in order to be able to compute all the other subsequent keys for the following periods of time. This means that an adversary who gains access to a secret key for a certain period of time, would be able to compute the secret keys for all periods of time (including previous ones).

The main difficulty encountered when trying to add forward security to the ABS derived from [Tsa17] is then the way to deal with the trapdoors for each of the time periods. This includes the trapdoors considered in the ABS construction as well as in the simulation. Moreover this modification induces a new

time parameter t, that has to be handled in the unforgeability proof. Indeed, the construction of [Tsa17] has been designed to only consider a fixed matrix $\mathbf{A}$ and a vector of matrices $\overrightarrow{\mathbf{A}}$ linked to the attribute to generate and verify signatures. But now we add $\log t$ additional matrices in order to integrate the time parameter, in a similar way to [LNWX19]. This transformation implies that the secret keys have to be modified according to the time period considered. It means that a trapdoor update mechanism needs to be built from the trapdoor construction of Tsabary, using tools introduced in the bonsai tree mechanism [CHKP10], and the time component has to be dealt with in the different queries from the simulation-based proof.

Finally, as we apply forward-secure property to an attribute-based construction in our case, we also have to handle an additional component which is the attribute. A naive adaptation from the transformation of [LNWX19] (on a group signature) to our construction (an attribute based signature) would not be secure. Indeed, we have to deal with two types of trapdoor: the trapdoor inherent to the ABS construction derived from [Tsa17], and the trapdoors given by the matrices linked to the time parameter. In the security proof of the ABS scheme, we need to simulate these two types of trapdoors according to each other, and according to the time period considered, in order to be able to answer all the queries of an attacker. At the same time, we expect all these trapdoors to vanish when the forgery of the attacker is outputted, in order to be able to conclude the simulation and then to argue about the security reduction getting a solution to a hard problem.

Related work. The only previous work on forward-secure group signature schemes from lattices is the work of [LNWX19] in the random oracle model using NIZK achieving signature size $\tilde{O}(\lambda(\log\ N + \log\ T))$ and group public key size $\tilde{O}(\lambda^2(\log\ N + \log\ T))$. Our scheme is constant in the number of group members and logarithmic in the number of time periods i.e. $\tilde{O}(\lambda \log\ T)$ and group public key size $\tilde{O}(\lambda^2 \log\ T)$. Their scheme satisfies full-anonymity and forward-secure traceability under SIS and LWE hardness.

Open problems. One open problem would be to achieve a group signature scheme with the same properties without relying on complexity leveraging (that we need to employ in the underlying ABS). Another open problem would be to upgrade the anonymity property from selfless anonymity to full anonymity.

2 Preliminaries

2.1 Lattices and trapdoors

In this paper we use several values defined as follows: λ is the security parameter and n, m and $q \geq 2$ are integers such that $n = \text{poly}(\lambda)$ and $m \geq n\lceil \log\ q \rceil$. The discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^m,\tau}$ over $\mathbb{Z}^m$ with parameter τ is the distribution where the probability of all $\mathbf{x}$ is proportional to $e^{-\pi\|\mathbf{x}\|/\tau^2}$. The norm of a

matrix $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_m] \in \mathbb{Z}_q^{n \times m}$, is denoted $\|\mathbf{A}\| = \max_j \|\mathbf{a}_j\|, j \in [m]$, and it is the maximum of the Euclidean norm of its vectors.

Lattices. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{u} \in \mathbb{Z}_q^n$ that admits a solution to the equation $\mathbf{A} \cdot \mathbf{x} = \mathbf{u} \bmod q$, define the m-dimensional lattice: $\Lambda^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \mathbf{x} = \mathbf{0} \bmod q\} \subseteq \mathbb{Z}^m$, and the coset $\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \mathbf{x} = \mathbf{u} \bmod q\}$.

We briefly remind the SIS assumption and its hardness.

Definition 1 ($\mathrm{SIS}_{n,q,B,m}$)**.** *Given a uniformly chosen matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, find nonzero integer vector $\mathbf{s} \in \mathbb{Z}^m$ such that $\|\mathbf{s}\|_\infty \leq B$ and $\mathbf{A} \cdot \mathbf{s} = \mathbf{0} \mod q$. $\mathrm{SIS}_{n,q,B,m}$ is hard if for any adversary $\mathcal{A}$, the probability to solve* SIS *is negligible, i.e. it is bounded by negl(λ). $\mathrm{SIS}_{n,q,B,m}$ is sub-exponentially hard if the probability to solve* SIS *is bounded by $2^{-O(n^\epsilon)} \cdot negl(\lambda)$ for some constant $0 < \epsilon < 1$.*

Trapdoors. For all $\mathbf{v} \in \mathbb{Z}_q^n$, $\mathbf{A}_{\gamma_0}^{-1}(\mathbf{v})$ is the random variable with discrete gaussian distribution $D_{\mathbb{Z}^m, \gamma_0}$ conditioned on $\mathbf{A} \cdot \mathbf{A}_{\gamma_0}^{-1}(\mathbf{v}) = \mathbf{v} \bmod q$. A γ_0-trapdoor for $\mathbf{A}$ allows a procedure that can sample from $\mathbf{A}_{\gamma_0}^{-1}(\mathbf{v})$ in time $\mathrm{poly}(n, m, \log q)$ for any $\mathbf{v} \in \mathbb{Z}_q^n$. By overloading notation we denote a γ_0-trapdoor for $\mathbf{A}$ by $\mathbf{A}_{\gamma_0}^{-1}$.

We define the gadget matrix $\mathbf{G}$ based on the vector $\mathbf{g} \in \mathbb{Z}_q^k$ whose entries are the power of two $\mathbf{g}^t := [1 \quad 2 \quad 4 \quad \cdots \quad 2^{k-1}]$ and $k = \lceil \log q \rceil$. The matrix $\mathbf{G}$ is the diagonal concatenation of $\mathbf{g}$ n times, i.e. $\mathbf{G} = \mathbf{g} \otimes \mathbf{I}_n \in \mathbb{Z}_q^{n \times nk}$.

Lemma 1 (Trapdoor generation [Ajt96,MP12]). *There exists an efficient procedure, that we call* $\mathrm{TrapGen}(1^n, 1^m, q)$*, with an efficiently computable value $m_0 = O(n \log q)$ such that for all $m \geqslant m_0$ outputs a pair $(\mathbf{A}, \mathbf{A}_{\gamma_0}^{-1})$, where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is at negligible distance from uniform and $\mathbf{A}_{\gamma_0}^{-1}$ is a γ_0-trapdoor for $\mathbf{A}$ with $\gamma_0 = O(\sqrt{n \log q \log n})$.*

Lemma 2 (Leftover Hash Lemma [HILL99]). *Let $m, n, q \geqslant 1$ be integers such that $m \geqslant 4n \log q$ and q prime. Let $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$, $\mathbf{r} \xleftarrow{\$} \{0,1\}^m$, then $(\mathbf{A}, \mathbf{Ar})$ is at negligible statistical distance from uniform distribution on $\mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^n$.*

2.2 Delegation functions

During different time periods, a signer will need to delegate some lattice trapdoor from a previous period to a next one. We make use of the following lemmas.

Lemma 3 (Trapdoor extension [ABB10,MP12]). *Let $\in \mathbb{Z}_q^{n \times m}$ be a matrix with trapdoor $\mathbf{M}_\gamma^{-1}$ and $\mathbf{N} \in \mathbb{Z}_q^{n \times p}$ a matrix such that $\mathbf{M} = \mathbf{NS}$ mod q where $\mathbf{S} \in \mathbb{Z}_q^{p \times m}$ with $s_1(\mathbf{S})$ its largest singular value. Then we can use $(\mathbf{M}_\gamma^{-1}, \mathbf{S})$ to sample from $\mathbf{N}_{\gamma'}^{-1}$ for any $\gamma' \geq \gamma \cdot s_1(\mathbf{S})$.*

Lemma 4 ([CHKP10, Lemma 3.2]). *There is a deterministic polynomial-time algorithm* ExtBasis *with the following properties: given an arbitrary $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ whose columns generate the entire group $\mathbb{Z}_q^n$, an arbitrary basis $\mathbf{S} \in \mathbb{Z}^{m \times m}$ of $\Lambda^{\perp}(\mathbf{A})$, and an arbitrary $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$,* $\mathrm{ExtBasis}(\mathbf{S}, \mathbf{A}' = \mathbf{A} \| \bar{\mathbf{A}})$ *outputs a basis $\mathbf{S}'$ of $\Lambda^{\perp}(\mathbf{A}') \subseteq \mathbb{Z}^{m+\bar{m}}$ such that $\|\tilde{\mathbf{S}}'\| = \|\tilde{\mathbf{S}}\|$. Moreover the same holds even for any permutation of the columns of $\mathbf{A}'$.*

There exists a function RandBasis developed by [CHKP10], which verifies the following lemma:

Lemma 5 ([CHKP10, Lemma 3.3]). *Let* $\mathbf{S}$ *be a basis of a* m*-dimensional integer lattice* Λ *and a parameter* $s \geq \|\tilde{\mathbf{S}}\| \cdot \omega(\sqrt{\log n})$*. The algorithm* $\mathrm{RandBasis}(\mathbf{S}, s)$ *outputs a new basis* $\mathbf{S}'$ *of* Λ *such that, with overwhelming probability,* $\mathbf{S}'$ *verifies* $\|\mathbf{S}'\| \leqslant s \cdot \sqrt{m}$*. Moreover, for any two basis* $\mathbf{S}_0, \mathbf{S}_1$ *of the same lattice and any* $s \geqslant \max\{\|\tilde{\mathbf{S}}_0\|, \|\tilde{\mathbf{S}}_1\|\} \cdot \omega(\sqrt{\log n})$*, the outputs of* $\mathrm{RandBasis}(\mathbf{S}_0, s)$ *and* $\mathrm{RandBasis}(\mathbf{S}_1, s)$ *are within* $\mathrm{negl}(n)$ *statistical distance.*

We further need an important property of lattice trapdoors ([ABB10],[MP12]):

Lemma 6. *For* $\mathbf{A} \in \mathbb{Z}_q^{n \times p}$ *and* $\mathbf{R} \in \mathbb{Z}_q^{p \times m}$ *with* $m = n\lceil \log q \rceil$*, one can compute* $[\mathbf{A} \| \mathbf{A}\mathbf{R} + \mathbf{G}]_\gamma^{-1}$ *for* $\gamma = O(\sqrt{mp}\, \|\mathbf{R}\|_\infty)$*.*

2.3 Evaluation functions

In order to generate or check the validity of a signature, we need to execute some evaluation of a function with a set of lattices as input. The output of this evaluation is 1 if the function evaluated on an attribute x is not valid and 0 if the evaluation is correct. We use the notations and definition of the evaluation functions developed by Tsabary [Tsa17]. Moreover we denote $[x_1\mathbf{G}|\cdots|x_\ell\mathbf{G}]$ by $\mathbf{x} \otimes \mathbf{G}$ with $\mathbf{x} = (x_1, \cdots, x_\ell) \in \{0,1\}^\ell$.

Theorem 1 ([Tsa17, Theorem 2.7]). *There exist efficient deterministic algorithms* EvalF *and* EvalFX *such that for all* $n, q, \ell \in \mathbb{N}$*,* $m = n\lceil \log q \rceil$*, and for any sequence of matrices* $\overrightarrow{\mathbf{A}} = (\mathbf{A}_1, \cdots, \mathbf{A}_l) \in (\mathbb{Z}_q^{n \times m})^\ell$*, for any depth* d *boolean circuit* $f : \{0,1\}^\ell \to \{0,1\}$ *and for every* $\mathbf{x} = (x_1, \cdots, x_\ell) \in \{0,1\}^\ell$*, the outputs* $\mathbf{H}_f = \mathsf{EvalF}(f, \overrightarrow{\mathbf{A}})$*, and* $\mathbf{H}_{f,\mathbf{x}} = \mathsf{EvalFX}(f, \mathbf{x}, \overrightarrow{\mathbf{A}})$ *are in* $\mathbb{Z}^{(\ell m) \times m}$ *and it holds that* $\|\mathbf{H}_f\|_\infty, \|\mathbf{H}_{f,\mathbf{x}}\|_\infty \leqslant (2m)^d$ *and* $(\overrightarrow{\mathbf{A}} - \mathbf{x} \otimes \mathbf{G}) \cdot \mathbf{H}_{f,\mathbf{x}} = \overrightarrow{\mathbf{A}} \cdot \mathbf{H}_f - f(\mathbf{x})\mathbf{G} \mod q$*.*

As in [KY19], we employ **secret key encryption** (SKE) and **one-time signature** (OTS), both from lattices, in order to build our group signature scheme. We use the SKE scheme based on LWE from [KY19], which is a secret key variant of [Reg05] and the OTS scheme from [Moh10]

2.4 Secret Key Encryption (SKE) from LWE

In this subsection we present the secret key encryption scheme from LWE defined by [KY19]. Recall that we need SKE for our both proposals for group signature scheme. The SKE scheme described below is IND-CPA secure and has a decryption circuit with $O(\log \lambda)$-depth. We mention that for the security of the group signature schemes that we presented in this paper, we need IND-CCA security which can be achieved from IND-CPA security and message authentication codes (MAC) scheme by a generic construction as explained in [KY19].

The SKE scheme below is a secret key variant of the Regev's scheme [Reg05]. The parameters of the scheme are the following: λ is the security parameter, l

is the dimension of the message space $\mathcal{M} = \{0,1\}^l$, $n = \text{poly}(\lambda)$, q is a prime polynomially bounded with $q \geq 24n + 2$ and $m = \lambda + n\lceil \log q \rceil$

$\mathsf{SKE.Setup}(1^\lambda)$ Set the dimensions n and m of the matrix and the modulus q. Output $\mathsf{pp} = (n, m, q)$.

$\mathsf{SKE.Gen}(\mathsf{pp})$ Sample two matrices $\mathbf{S}_0 \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ and $\mathbf{S}_1 \xleftarrow{\$} \mathbb{Z}_q^{n\times l}$. Output the secret key $K = (\mathbf{S}_0, \mathbf{S}_1)$

$\mathsf{SKE.Enc}(K, M)$ Parse K as $(\mathbf{S}_0, \mathbf{S}_1)$. Sample $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n \setminus \{\mathbf{0}\}, \mathbf{x}_0 \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^m, 3\sqrt{n}}$ and $\mathbf{x}_1 \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^l, 3\sqrt{n}}$. Compute

$$\mathbf{c}_0^\top := \mathbf{a}^\top \mathbf{S}_0 + \mathbf{x}_0^\top \quad \mathbf{c}_1^\top := \mathbf{a}^\top \mathbf{S}_1 + \mathbf{x}_1^\top + \lceil q/2 \rceil \cdot M$$

where $M \in \{0,1\}^l$ is considered as a row vector. Finally, output the ciphertext $\mathsf{ct} := (\mathbf{a}, \mathbf{c}_0, \mathbf{c}_1)$

$\mathsf{SKE.Dec}(K, \mathsf{ct})$ Parse ciphertext as $\mathsf{ct} = (\mathbf{a}, \mathbf{c}_0, \mathbf{c}_1)$ and check its validity. Compute $\mathbf{v}_0^\top := \mathbf{c}_0^\top - \mathbf{a}^\top \mathbf{S}_0$ and $\mathbf{v}_1^\top := \mathbf{c}_1^\top - \mathbf{a}^\top \mathbf{S}_1$. If $\mathbf{v}_0 \notin [-3n, 3n]^m$, output $\mathsf{Invalid}$. Otherwise, recover $M_i \in \{0,1\}^l$ for $i \in [l]$ as follows: if the i-th coefficient of $\mathbf{v}_1$ is in $[-3n, 3n]$ then M_i is 0, otherwise, M_i is 1.

Correctness. This follows since we have $\|\mathbf{x}_0\|_\infty$, $\|\mathbf{x}_1\|_\infty \leq 3n$ with probability 1 and $q \geq 24n + 2$.

2.5 One-Time Signature (OTS) scheme

A one-time signature scheme is defined by the following algorithms:

$\mathsf{OTS.KeyGen}(1^\lambda)$ is a randomized algorithm taking as input a security parameter 1^λ and outputs a verification key ovk and a signing key osk.

$\mathsf{OTS.Sign}(\mathsf{osk}, M)$ takes as input a secret key osk and a message M and outputs a signature σ.

$\mathsf{OTS.Verify}(\mathsf{ovk}, \sigma, M)$ takes as input a verification public key ovk, a message M and a signature σ and outputs Valid or $\mathsf{Invalid}$.

Correctness. For all λ, $(\mathsf{ovk}, \mathsf{osk}) \in \mathsf{OTS.KeyGen}(1^\lambda)$, M in the message space and $\sigma \in \mathsf{OTS.Sign}(\mathsf{osk}, \mathsf{M})$,

$\mathsf{OTS.Verify}(\mathsf{ovk}, \sigma, M) = \mathsf{Valid}$ holds.

The security notion required for OTS scheme is the classical *strong unforgeability* with the difference that the adversary is allowed to make a single signing query instead of polynomially many. We define it using a game model between a challenger and an attacker $\mathcal{A}$ to define this security notion.

Setup: At the beginning of the game, the challenger runs $\mathsf{OTS.KeyGen}(1^{\check{}}) \to (\mathsf{ovk}, \mathsf{osk})$ and gives 1^λ and ovk to $\mathcal{A}$.

Signing Query: During the game $\mathcal{A}$ can perform a single signing query. When receiving M, the challenger runs $\mathsf{OTS.Sign}(\mathsf{osk}, M) \to \sigma$ and returns (M, σ) to $\mathcal{A}$.

Forgery: Eventually $\mathcal{A}$ outputs (M^*, σ^*) as a forgery. Then $\mathcal{A}$ wins the game if $\mathsf{OTS.Verify}(\mathsf{ovk}, \sigma^*, M^*) = \mathsf{Valid}$ and $(M, \sigma) \neq (M^*, \sigma^*)$

We then define the advantage of the attacker $\mathcal{A}$ as the probability that $\mathcal{A}$ wins the above game.

We say that an OTS scheme is strongly unforgeable, if for all PPT adversary, the advantage in the above game is negligible.

2.6 Admissible hash functions

Admissible hash functions represent a family of hash functions introduced in [BB04], which allows to separate the input space into two sets, the challenge set and the controlled set. In practice, in a simulation-based game, a simulator owning a dummy key can answer to queries in the controlled set but not in the challenge set, and the adversary is expected to make his forgery in the challenge set, allowing the simulator to solve a hard problem.

We fit in the definition of admissible hash functions given in [KY19].

Intuitively, WldCmp is a string comparison function with wildcards which takes as input three strings y, z, w and compares z and w only at those points where $y_i = 1$.

Definition 2 ([KY19, definition 1]). *Let $\ell := \ell(\lambda)$ and $\ell' := \ell'(\lambda)$ be some polynomials. We define the function* $\mathrm{WldCmp} : \{0,1\}^{\ell} \times \{0,1\}^{\ell} \times \{0,1\}^{\ell} \to \{0,1\}$ *as*

$$\mathrm{WldCmp}(y, z, w) = 0 \Leftrightarrow \forall i \in [\ell], ((y_i = 0) \vee (z_i = w_i)),$$

where y_i, z_i and w_i denote the i-th bit of y, z and w respectively. Let $\{\mathrm{H}_\lambda : \{0,1\}^{\ell'(\lambda)} \to \{0,1\}^{\ell(\lambda)}\}_{\lambda \in \mathbb{N}}$ be a family of admissible hash functions if there exists an efficient algorithm AdmSmp *that takes as input 1^λ and $Q \in \mathbb{N}$ and outputs $(y, z) \in \{0,1\}^{\ell} \times \{0,1\}^{\ell}$ such that for every polynomial $Q(\lambda)$ and all $X^*, X^{(1)}, \cdots, X^{(Q)} \in \{0,1\}^{\ell'(\lambda)}$ with $X^* \notin \{X^{(1)}, \cdots, X^{(Q)}\}$, we have* $\Pr_{(y,z)}[\mathrm{WldCmp}(y, z, \mathrm{H}(X^*)) = 0 \wedge (\mathrm{WldCmp}(y, z, \mathrm{H}(X^{(j)})) = 1 \ \forall j \in [Q])] \geqslant \Delta_Q(\lambda)$, *for a noticeable function $\Delta_Q(\lambda)$, where the probability above is taken over the choice of $(y, z) \xleftarrow{\$} \mathrm{AdmSmp}(1^\lambda, Q)$.*

3 Forward-Secure Indexed Attribute-Based Signature scheme from lattices

As already explained in the introduction, we replace the ABS scheme in the general construction of [KY19] with a forward-secure indexed ABS. We start by giving the definition and the security requirements of a forward-secure indexed attribute based signature. We note that the ABS scheme supports multiple users since it is designed as a building block for group signature scheme.

The starting point of our scheme is the constrained signature of [Tsa17]. We first adapt it into an indexed attribute-based signature, by including an index

i into the attribute x, following the idea of [KY19]. Moreover we extend this construction to a forward-secure attribute-based signature scheme, by applying a transformation similar to [LNWX19]. The idea of this transformation is that we consider a pair of matrices $\mathbf{T}_j^b, b \in \{0,1\}$ for every bit j of the time period t considered. Then by concatenating these matrices $\mathbf{T}_j^b$ to the public key of [Tsa17], we can include a time period t into the verification key and the signatures. The technical difficulty that arises when using this transformation into the Tsabary's construction is simulating the secret keys for each period of time and for each user, without possessing the master secret key. This can be done by using "dummy" secret keys which vanish when the signature is made for an identity and a time period chosen selectively by the adversary at the beginning of the game, allowing the simulator to solve a hard problem (which is the SIS problem). We then get a new forward-secure attribute-based signature scheme which is independent of the number of users N, and only logarithmic on the total number of periods T.

3.1 Framework and security properties

We denote $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ the set of circuits with domain $\{0,1\}^{k(\lambda)}$ and range $\{0,1\}$. We bound the size of every circuit in $\{\mathcal{C}_\lambda\}$ by $k_c = \text{poly}(\lambda)$. We also denote the space of messages as $\{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$, for which we bound the size elements by $k_m = \text{poly}(\lambda)$. Usually we simplify notation and just denote these spaces $\mathcal{C}$ and $\mathcal{M}$. We then define the forward-secure indexed attribute-based signature scheme for the circuit class $\mathcal{C}$:

Definition 3. *A forward-secure indexed attribute-based signature (FSI-ABS) scheme consists of the following algorithms:*

- $\mathsf{ABS.Setup}(1^\lambda, 1^N, 1^T)$ *The setup algorithm takes as input λ the security parameter, N the size of the index space and T the number of time periods, given in unary form, and it outputs a master public key* mpk *and a master secret key* msk.
- $\mathsf{ABS.KeyGen}(\mathsf{msk}, i, x_i)$ *The key generation algorithm takes as input the master secret key* msk*, an index $i \in [N]$ and the attribute $x_i \in \{0,1\}^k$. It outputs $\mathsf{sk}_{x_i,0}$, the initial secret key associated to x_i.*
- $\mathsf{ABS.KeyUpdate}(\mathsf{mpk}, i, \mathsf{sk}_{x_i,t}, t+1)$ *The key update algorithm takes as input the master secret key* msk*, an index of an user i as well as its secret key for the time t,* $\mathsf{sk}_{x_i,t}$*. It updates this key* $\mathsf{sk}_{x_i,t}$ *for the next time period $t+1$ and outputs* $\mathsf{sk}_{x_i,t+1}$.
- $\mathsf{ABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{x_i,t}, C, M, t)$ *The signing algorithm takes as input the master public key* mpk*, a secret key* $\mathsf{sk}_{x_i,t}$ *for the current period of time t, a circuit $C \in \mathcal{C}_\lambda$, a message $M \in \mathcal{M}_\lambda$ and a time period t and it outputs an attribute-based signature σ if $C(x_i) = 0$.*
- $\mathsf{ABS.Verify}(\mathsf{mpk}, C, M, \sigma, t)$ *The verification algorithm takes as input the master public key* mpk*, the circuit $\mathcal{C}$, the message M, the attribute-based signature σ and the time period t. This algorithm outputs* Valid *if the signature σ is valid for the time period t and* $\mathsf{Invalid}$ *otherwise.*

For a FSI-ABS scheme, we require *correctness* and two security properties: *perfect-privacy* and *forward-secure policy-selective unforgeability*. Perfect privacy captures the idea that the attribute used to sign a message must remain anonymous. The unforgeability property says that even if users collude they can not forge a signature on a message associated with a policy if none of the attributes satisfies the policy.We note that we can not achieve selective unforgeability directly, but we start from no-signing-query and apply a transformation using admissible hash functions to obtain selective unforgeability. We explain this in more detail at the end of this section.

Correctness. We require that for all $\lambda, N \in \mathsf{poly}(\lambda), T \in \mathbb{N}, t \in [T]$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{ABS.Setup}(1^\lambda, 1^N, 1^T)$, $i \in [N]$, $x_i \in \{0,1\}^k, C \in \mathcal{C}_\lambda$ such that $C(x_i) = 0$, $M \in \mathcal{M}_\lambda, \mathsf{sk}_{x_i,0} \leftarrow \mathsf{ABS.KeyGen}(\mathsf{msk}, i, x_i), \mathsf{sk}_{x_i,t} \leftarrow \mathsf{ABS.KeyUpdate}(\mathsf{mpk}, i, \mathsf{sk}_{x_i,t-1}, t)$, $\sigma \leftarrow \mathsf{ABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{x_i,t}, C, M, t)$, we have that $\mathsf{ABS.Verify}(\mathsf{mpk}, C, M, \sigma, t) = \mathsf{Valid}$.

Perfect privacy. A FSI-ABS scheme provides **perfect privacy** if for all $\lambda, N \in \mathrm{poly}(\lambda)$, $T \in \mathbb{N}$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{ABS.Setup}(1^\lambda, 1^N, 1^T)$, $x_0, x_1 \in \{0,1\}^k$, $i_0, i_1 \in [N]$, $C \in \mathcal{C}_\lambda$, $t \in [T]$, $C(x_0) = C(x_1) = 0$, $M \in \mathcal{M}_\lambda$,
$\mathsf{sk}_{x_b,0} \leftarrow \mathsf{ABS.KeyGen}(\mathsf{msk}, i_b, x_b)$ and $\mathsf{sk}_{x_b,t} \leftarrow \mathsf{ABS.KeyUpdate}(\mathsf{mpk}, b, \mathsf{sk}_{x_b,t-1}, t)$, the distributions $\mathsf{ABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{x_0,t}, C, M, t)$ and $\mathsf{ABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{x_1,t}, C, M, t)$ are indistinguishable.

Forward-secure policy-selective unforgeability. We define the **forward-secure policy-selective unforgeability** for an indexed attribute-based signature scheme following the framework from [YLH+12]. We use a game model between a challenger and an attacker $\mathcal{A}$ to define this security notion.

Setup: At the beginning of the game, the adversary $\mathcal{A}$ is given $1^\lambda, 1^N, 1^T$ as input. It then sends to the challenger the tuple (C^*, M^*, t^*) consisting of a circuit, a message and a time period for which he is going to forge a signature. The challenger gets $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{ABS.Setup}(1^\lambda, 1^N, 1^T)$. It gives mpk to $\mathcal{A}$. At the start of each time period $t \in [T]$, the challenger announces the beginning of t to $\mathcal{A}$. During current time period t, the challenger responds to $\mathcal{A}$'s queries as follows:

Key Queries: $\mathcal{A}$ sends (i, x_i, t) to the challenger and gets back $\mathsf{sk}_{x_i,t}$.

Signing Queries: $\mathcal{A}$ can perform some signing queries to the challenger during the game.
If $\mathcal{A}$ queries (C, M, t, i), with $M \in \mathcal{M}, C \in \mathcal{C}, i \in [N]$ and $C(x_i) = 0$, the challenger generates $\sigma \leftarrow \mathsf{ABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{x_i,t}, C, M, t)$ and sends it to $\mathcal{A}$.

Forgery: Eventually $\mathcal{A}$ outputs $(C^*, M^*, \sigma^*, t^*)$ as a forgery. Then $\mathcal{A}$ wins the game if:

1. $C^* \in \mathcal{C}$,
2. $(C^*, M^*, t^*, \cdot)$ was not queried in a Signing query,
3. $\mathsf{ABS.Verify}(\mathsf{mpk}, C^*, M^*, \sigma^*, t^*) = \mathsf{Valid}$,
4. $C^*(x_i) = 1$ for any key queried by $\mathcal{A}$ respective to t and x_i where $t \leq t^*$.

We then define the advantage of the attacker $\mathcal{A}$ against forward-secure policy-selective unforgeability as the probability that $\mathcal{A}$ wins the above game. We say that a scheme satisfies the forward-secure policy-selective unforgeability property, if for all PPT adversary, the advantage in the above game is negligible.

In our ABS construction, we make use of two different flavours of unforgeability as in [KY19]:

- **No-signing-query unforgeability:** The game model of the no-signing-query unforgeability is the same as policy-selective unforgeability, but the attacker can not perform signature queries.
- **Adaptive unforgeability:** The game model of the adaptive unforgeability is the same as the policy-selective unforgeability, but the attacker is not asked anymore to give the tuple (C^*, M^*, t^*) at the beginning of the game, he can rather choose it adaptively during the game.

3.2 Construction of FSI-ABS scheme from lattices

We adapt the constrained signature developed by Tsabary [Tsa17] to a forward-secure attribute-based signature scheme. As explained by Katsumata and Yamada [KY19], the signature scheme of Tsabary is not an attribute-based signature but a constrained signature. It means that in the constrained signature, a user does not sign a circuit but an attribute. Then the role of the attribute and the circuit are exchanged compared to an actual attribute-based signature scheme. However, as explained in [KY19], we can turn a constrained signature into an attribute-based signature: we consider a constraint space composed of all d-depth bounded circuit $\mathcal{F}_d = \{f : \{0,1\}^\ell \to \{0,1\}\}$, with $\ell = \text{poly}(\lambda)$, then a constraint f can be seen as a universal circuit $U(\cdot,\cdot,x)$ (that we denote $U_x(\cdot,\cdot)$), which takes as input the circuit-message pair (C, M) (seen as a string of size ℓ).

Our contribution is to build a forward-secure attribute-based signature scheme meaning that the lifetime of the scheme is divided into $T = 2^d$ discrete periods. To represent the time periods we use a binary tree, then each time period t is associated with a leaf $\mathsf{Bin}(t)$. Following [BSSW06], for $j \in [d+1]$, we define a time period's "second sibling at depth j". Intuitively, it corresponds to the right neighbour at depth j of each node on the path from the root to the leaf $\mathsf{Bin}(t)$.

$$\mathsf{Sibling}(j,t) = \left\{ \begin{array}{ll} (1) & \text{if } j = 1 \text{ and } \mathsf{Bin}(t)[j] = 0 \\ (\mathsf{Bin}(t)[1], ..., \mathsf{Bin}(t)[j-1], 1) & \text{if } 1 < j \leq d \text{ and } \mathsf{Bin}(t)[j] = 0 \\ \perp & \text{if } 1 \leqslant j \leq d \text{ and } \mathsf{Bin}(t)[j] = 1 \\ \mathsf{Bin}(t) & \text{if } j = d+1 \end{array} \right\}.$$

We also define node set $\text{Nodes}_{(t \to T-1)}$ to be $\{\mathsf{Sibling}(1,t), ..., \mathsf{Sibling}(d+1,t)\}$. The goal of this set is to uniquely define the path to each leaf of the tree.

We consider also a function called bitstr which takes as input a message-circuit pair (C, M) and which outputs its input seen as a string of bits. Then bitstr : $\{0,1\}^{k_c} \times \{0,1\}^{k_m} \mapsto \{0,1\}^\ell$, such that $\mathsf{bitstr}(C, M) = \{C_1, \cdots, C_{k_c}, M_1, \cdots, M_{k_m}\}$.

Selection of parameters Given the security parameter λ, the parameters m_0, p, γ_0 and τ_s are chosen according to $\mathsf{TrapGen}$ algorithm, $T = 2^d$ is chosen as a power of 2, for $d \in \mathbb{N}$, and is the number of time periods considered, and ℓ is the size of input of the circuit. We choose parameters τ_u and B by referring to Theorem 1. Finally s_j is dictated also by Lemma 3. Then we set:

- $m = 4n\lceil \log q \rceil$, $m_0 = O(n \log q) \geqslant 4n \log q$,
- $p = \max\{m_0, (n+1)\lceil \log q \rceil + 2\lambda\}$,
- $\gamma_0 = O(\sqrt{n\lceil \log q \rceil \log n})$,
- $\tau_s = \max\{\sqrt{p} \cdot \ell \cdot 2^d m^{1.5+d}, \gamma_0\}$,
- $\tau_u = \tau_s \cdot \sqrt{\ell} \cdot 2^d m^{0.5+d}$,
- $B = \tau_u \sqrt{(1+d) \cdot p + \ell \cdot m}$,
- $s_j = \mathcal{O}(\sqrt{nd \log q})^{j+1} \cdot \omega(\sqrt{\log n})^{j+1}$ for $j \in [d]$.

ABS.Setup$(1^\lambda, 1^N, 1^T)$ On input the security parameter 1^λ, 1^N where N is the number of indexes $i \in [N]$ and 1^T where T is the number of time periods $T = 2^d$ for some $d \in \mathbb{N}$, it sets the parameters n, m, p, q, γ_0 to be polynomial in λ. Then, it generates:

- uniform matrix $\overrightarrow{\mathbf{A}} = [\mathbf{A}_1 \| ... \| \mathbf{A}_\ell] \xleftarrow{\$} \mathbb{Z}_q^{n \times \ell m}$,
- $(\mathbf{A}, \mathbf{A}_{\gamma_0}^{-1}) \leftarrow \mathsf{TrapGen}(1^n, 1^p, q)$, with $\mathbf{A} \in \mathbb{Z}_q^{n \times p}$ and $\mathbf{A}_{\gamma_0}^{-1}$ its trapdoor,
- $2d$ matrices $\mathbf{T}_j^b \xleftarrow{\$} \mathbb{Z}_q^{n \times p}$ for all $j \in [d]$ and $b \in \{0,1\}$.

The algorithm outputs: $\mathsf{mpk} = (\mathbf{A}, \overrightarrow{\mathbf{A}}, \{\mathbf{T}_j^b\}_{j \in [d], b \in \{0,1\}})$ and $\mathsf{msk} = (\mathbf{A}_{\gamma_0}^{-1})$.

ABS.KeyGen(msk, i, x_i) On input the master secret key msk, the index $i \in [N]$ and the attribute $x_i \in \{0,1\}^k$, it computes U_{x_i}, $\mathbf{H}_{U_{x_i}} = \mathsf{EvalF}(U_{x_i}, \overrightarrow{\mathbf{A}}) \in$

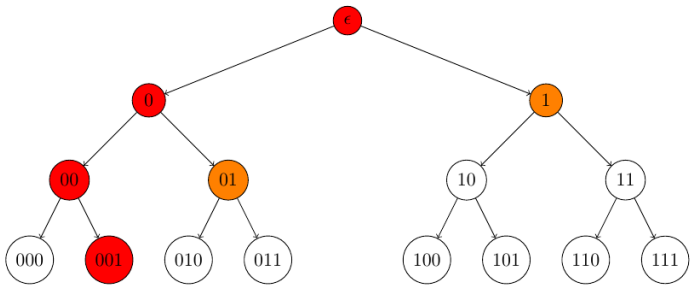

Fig. 1. A binary tree with time periods $\mathrm{T} = 2^3$. In order to fill the set $\mathrm{Nodes}_{(t \to T-1)}$ we begin with the leaf $\mathrm{Bin}(t)$ that we add in the set $\mathrm{Nodes}_{(t \to T-1)}$, together with its sibling (which is its right neighbour), if it exists. Then recursively, we go up in the tree to the parent of the node considered (coloured in red), and we add its sibling (coloured in orange) to the set $\mathrm{Nodes}_{(t \to T-1)}$ (still if it exists). We keep going this way, until we reach the root of the binary tree. We stop then and output the corresponding list $\mathrm{Nodes}_{(t \to T-1)}$. On the path from node ϵ to the leaf node (001) we then have $\mathrm{Nodes}_{(1 \to 7)} = \{(1), (01), \perp, (001)\}$.

$\mathbb{Z}_q^{\ell m \times m}$ as defined in Theorem 1 and $\mathbf{A}_{x_i} = \overrightarrow{\mathbf{A}} \cdot \mathbf{H}_{U_{x_i}} \in \mathbb{Z}_q^{n \times m}$. Then, it uses $\mathbf{A}_{\gamma_0}^{-1}$ to compute $\mathbf{R}_{x_i} = [\mathbf{A} \| \mathbf{A}_{x_i}]^{-1}$. Then it determines the set $\mathsf{Nodes}_{(0 \to T-1)}$ and for $z \in \mathsf{Nodes}_{(0 \to T-1)}$:

- if $z = \perp$, set $\mathsf{sk}_{x_i}[z] = \perp$,
- else it denotes d_z as the bit-length of z, with $d_z \leqslant d$, and computes the matrix: $\mathbf{A}_{x_i,z} = [\mathbf{A} \| \mathbf{A}_{x_i} \| \mathbf{T}_1^{\mathsf{Bin}(z)[1]} \| \cdots \| \mathbf{T}_{d_z}^{\mathsf{Bin}(z)[d_z]}] \in \mathbb{Z}_q^{n \times ((d_z+1)p+m)}$, then it computes: $\mathbf{R}_{x_i,z} \leftarrow \mathrm{RandBasis}(\mathrm{ExtBasis}(\mathbf{R}_{x_i}, \mathbf{A}_{x_i,z}), s_{d_z})$, and set $\mathsf{sk}_{x_i}[z] = \mathbf{R}_{x_i,z}$,

Finally we get: $\mathsf{sk}_{x_i,0} = \{\mathsf{sk}_{x_i}[z], z \in \mathsf{Nodes}_{(0 \to T-1)}\}$.

ABS.KeyUpdate$(\mathsf{mpk}, i, \mathsf{sk}_{x_i,t}, t+1)$ First parse the set $\mathsf{sk}_{x_i,t} = \{\mathsf{sk}_{x_i}[z], z \in \mathsf{Nodes}_{(t \to T-1)}\}$ and determine the set $\mathsf{Nodes}_{(t+1 \to T-1)}$.

For $z' \in \mathsf{Nodes}_{(t+1 \to T-1)}$:

- if $z' = \perp$, set $\mathsf{sk}_{x_i}[z'] = \perp$.
- Otherwise, there exists exactly one $z \in \mathsf{Nodes}_{(t \to T-1)}$ which is a prefix of z' i.e. $z' = z \| y$. There are two possibilities here:
  1. if $z' = z$ then $\mathsf{sk}_{x_i}[z'] = \mathsf{sk}_{x_i}[z]$,
  2. if $z' = z \| y$ for some non-empty y, then z is an ancestor of z', and from $\mathsf{sk}_{x_i}[z] = \mathbf{R}_{x_i,z}$ it can delegate a basis $\mathbf{R}_{x_i,z'} \leftarrow \mathrm{RandBasis}(\mathrm{ExtBasis}(\mathbf{R}_{x_i,z}, \mathbf{A}_{x_i,z'}), s_{d_{z'}})$, and set $\mathsf{sk}_{x_i}[z'] = \mathbf{R}_{x_i,z'}$.

Finally output $\mathsf{sk}_{x_i,t+1} = \{\mathsf{sk}_{x_i}[z'], z' \in \mathsf{Nodes}_{(t+1 \to T-1)}\}$.

ABS.Sign$(\mathsf{mpk}, \mathsf{sk}_{x_i,t}, C, M, t)$ First compute $\mathbf{x} = \mathsf{bitstr}(C, M)$. If $U_{x_i}(\mathbf{x}) = C(x_i) \neq 0$ output $\perp$. Otherwise, first compute $\mathbf{H}_{U_{x_i},\mathbf{x}} = \mathsf{EvalFX}(U_{x_i}, \mathbf{x}, \overrightarrow{\mathbf{A}}) \in \mathbb{Z}_q^{\ell m \times m}$, as defined in Theorem 1, such that $(\overrightarrow{\mathbf{A}} - \mathbf{x} \otimes \mathbf{G}) \cdot \mathbf{H}_{U_{x_i},\mathbf{x}} = \overrightarrow{\mathbf{A}} \cdot \mathbf{H}_{U_{x_i}} - U_{x_i}(\mathbf{x})\mathbf{G} = \mathbf{A}_{x_i}$ as $U_{x_i}(\mathbf{x}) = 0$.

Compute $\overrightarrow{\mathbf{B}_t} = [\mathbf{A} \| \overrightarrow{\mathbf{A}} - \mathbf{x} \otimes \mathbf{G} \| \mathbf{T}_1^{\mathsf{Bin}(t)[1]} \| \cdots \| \mathbf{T}_d^{\mathsf{Bin}(t)[d]}] \in \mathbb{Z}_q^{n \times ((d+1)p+\ell m)}$, and

$$\mathbf{S}_i = \begin{bmatrix} \mathbf{I}_p & & & & \\ & \mathbf{H}_{U_{x_i},\mathbf{x}} & & & \\ & & \mathbf{I}_p & & \\ & & & \cdots & \\ & & & & \mathbf{I}_p \end{bmatrix} \in \mathbb{Z}_q^{((d+1)p+\ell m) \times ((d+1)p+m)}.$$

We then have $\overrightarrow{\mathbf{B}_t} \cdot \mathbf{S}_i = [\mathbf{A} \| \mathbf{A}_{x_i} \| \mathbf{T}_1^{\mathsf{Bin}(t)[1]} \| \cdots \| \mathbf{T}_d^{\mathsf{Bin}(t)[d]}] = \mathbf{A}_{x_i,t}$. Since $\mathsf{sk}_{x_i,t}$ contains a trapdoor for $\mathbf{A}_{x_i,t}$, we can apply the trapdoor extension from Lemma 3 to obtain $\mathbf{B}_{\tau_u}^{-1} = [\overrightarrow{\mathbf{B}_t}]_{\tau_u}^{-1} = [\mathbf{A} \| \overrightarrow{\mathbf{A}} - \mathbf{x} \otimes \mathbf{G} \| \mathbf{T}_1^{\mathsf{Bin}(t)[1]} \| \cdots \| \mathbf{T}_d^{\mathsf{Bin}(t)[d]}]_{\tau_u}^{-1}$, where $\mathbf{A} = \mathbf{A}_{x_i,t}$, $\mathbf{B} = \overrightarrow{\mathbf{B}_t}$ and $\mathbf{S} = \mathbf{S}_i$ using $\mathsf{sk}_{x_i,t} = [\mathbf{A}_{x_i,t}]_{\tau_s}^{-1}$.

Then the signer has a trapdoor for $\overrightarrow{\mathbf{B}_t}$ and he can compute $\sigma_{\mathbf{x},t} \xleftarrow{\$} \overrightarrow{\mathbf{B}_t}^{-1}(\mathbf{0})$.

ABS.Verify$(\mathsf{mpk}, C, M, \sigma_{\mathbf{x},t}, t)$. First, compute $\mathbf{x} = \mathsf{bitstr}(C, M)$ and then check that:

- $[\mathbf{A} \| \overrightarrow{\mathbf{A}} - \mathbf{x} \otimes \mathbf{G} \| \mathbf{T}_1^{\mathsf{Bin}(t)[1]} \| \cdots \| \mathbf{T}_d^{\mathsf{Bin}(t)[d]}] \cdot \sigma_{\mathbf{x},t} = \mathbf{0}$,
- $\|\sigma_{\mathbf{x},t}\|_\infty \leqslant B$.

If the verification passes, then output Valid, if not, output $\mathsf{Invalid}$.

Correctness. We fix an attribute $x_i \in \{0,1\}^k$ for user $i \in [N]$, a circuit $C \in \mathcal{C}_\lambda$ such that $C(x_i) = 0$, a time period t and a message M and let $\mathbf{x} = \mathsf{bitstr}(C, M)$. Consider $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{ABS.Setup}(1^\lambda, 1^N, 1^T)$ and
$\sigma_{\mathbf{x},t} \leftarrow \mathsf{ABS.Sign}(\mathsf{mpk}, \mathsf{ABS.KeyUpdate}(\mathsf{mpk}, i, \mathsf{sk}_{x_i,t-1}, t), C, M, t)$, since the signature $\sigma_{\mathbf{x},t} \neq \perp$ because of $C(x_i) = 0$, we have that $\sigma_{\mathbf{x},t} \in \mathbf{B}_{\tau_u}^{-1}(\mathbf{0}) = [\mathbf{A} \| \overrightarrow{\mathbf{A}} - \mathbf{x} \otimes \mathbf{G} \| \mathbf{T}_1^{\mathsf{Bin}(t)[1]} \| \cdots \| \mathbf{T}_d^{\mathsf{Bin}(t)[d]}]_{\tau_u}^{-1}(\mathbf{0})$ and
$[\mathbf{A} \| \overrightarrow{\mathbf{A}} - \mathbf{x} \otimes \mathbf{G} \| \mathbf{T}_1^{\mathsf{Bin}(t)[1]} \| \cdots \| \mathbf{T}_d^{\mathsf{Bin}(t)[d]}] \cdot \sigma_{\mathbf{x},t} = \mathbf{0}$. Moreover, we know from lattice trapdoor properties that samples from $\mathbf{B}_{\tau_u}^{-1}(\mathbf{0})$ have discrete Gaussian distribution over $\mathbb{Z}_q^{p+\ell m+dp}$ with parameter τ_u and, therefore,
$\|\sigma_{\mathbf{x},t}\|_\infty \leq \tau_u \sqrt{(1+d) \cdot p + \ell \cdot m} = B$ and $\mathsf{ABS.Verify}(\mathsf{mpk}, C, M, \sigma_{\mathbf{x},t}, t) = \mathsf{Valid}$.

3.3 Security proofs

Lemma 7. *Our ABS scheme is perfectly private.*

Proof. We consider the perfect privacy game defined in Section 3.1. We change the way each signature $\sigma_{\mathbf{x},b}$ is generated, with $\mathbf{x} = \mathsf{bitstr}(C, M), b \in \{0,1\}$. We use the trapdoor $\mathbf{A}_{\gamma_0}^{-1}$ to compute $[\mathbf{A} \| \overrightarrow{\mathbf{A}} - \mathbf{x} \otimes \mathbf{G} \| \mathbf{T}_1^{\mathsf{Bin}(t)[1]} \| \cdots \| \mathbf{T}_d^{\mathsf{Bin}(t)[d]}]_{\tau_u}^{-1}$ (we have $\tau_u > \gamma_0$). Then using this trapdoor, we compute $\sigma_{\mathbf{x},t,b} = [\mathbf{A} \| \overrightarrow{\mathbf{A}} - \mathbf{x} \otimes \mathbf{G} \| \mathbf{T}_1^{\mathsf{Bin}(t)[1]} \| \cdots \| \mathbf{T}_d^{\mathsf{Bin}(t)[d]}]_{\tau_u}^{-1}(\mathbf{0}), b \in \{0,1\}$. This modification does not change the distribution for each $\sigma_{\mathbf{x},t,b}, b \in \{0,1\}$, which means that this change is statistically indistinguishable. Now the signature generation is totally independent of the bit b. Then the ABS scheme is perfectly private. □

Lemma 8. *Our ABS satisfies forward-secure no-signing-query unforgeability assuming* $\mathsf{SIS}_{n,q,B',m'}$ *is hard, with* $B' = (\ell(m+d)+1)B$ *and* $m' = (d+1)p + \ell \cdot m$.

What we prove in this theorem is a weak property of unforgeability, where an attacker is prohibited to make signing queries. Indeed, we do not include the message to be signed in the different steps of the signature process and note that if the attacker would be able to perform some signature query on a circuit message pair (C, M) and get σ, he could just output the valid signature σ but on a pair (C, M') with $M' \neq M$ and win the unforgeability game.

However, in the context in which we intend to use the attribute-based signature, namely the group signature, this property of unforgeability is not enough. We note that they face the same problem in [KY19] and introduce a reduction from a (co-)selective unforgeable ABS to a no-signing-query ABS, using as a tool the admissible hash function defined in Definition 2. Adapting in the same way as their construction, we get a stronger unforgeability property, namely selective unforgeability.

With the above lemma and the no-signing to selective transformation of [KY19], we prove that the attribute-based signature derived from the constrained signature of Tsabary [Tsa17] is forward-secure policy-selective unforgeable where the adversary chooses its target circuit-message pair (C^*, M^*) for the forgery at the beginning of the game. But still this security notion is not enough for our

group signature scheme, we need adaptive security and we can only achieve it by utilizing complexity leveraging as suggested in [KY19]. We have to randomly guess (C^*, M^*) in the reduction from selective to adaptive security. Let us evaluate the reduction loss (as done in [KY19]): the length of the message M^* is bounded by $\text{poly}(\lambda)$ and a circuit $\mathcal{C}^*$ can be described by ovk and ct which can be seen as binary strings with length $\text{poly}(\lambda, \log N)$ inducing a reduction loss of $2^{-\text{poly}(\lambda, \log N)}$. To account for the loss in advantage we need to enlarge the dimension n of the scheme to be $\text{poly}(\lambda, \log N)^{1/\epsilon}$ where ϵ is some constant in $(0, 1)$ requiring subexponential hardness of the SIS problem. As mentioned in the introduction, co-selective unforgeability (where the adversary has to make all the key queries at the beginning of the game but he can choose the target policy adaptively) would be enough for our scheme but we can not achieve it directly since in the unforgeability game we need to target policy associated to the forgery to be chosen at the beginning of the game so that we can build the public matrix for which we solve the SIS problem.

Proof. To prove this lemma, we will show that for any PPT adversary $\mathcal{A}$ against the forward secure no-signing-query unforgeability of the ABS scheme with advantage ϵ, we can build a PPT algorithm $\mathcal{B}$ that solves SIS with probability at least $\epsilon - negl(\lambda)$. Therefore, by assuming the hardness of the SIS problem, we conclude that ϵ is negligible. The proof is done in a sequence of games where the first game is identical to the forward-secure no-signing-query unforgeability game defined in Section 3.1.

Game 0**:** The first game is the forward-secure no-signing-query unforgeability game between adversary $\mathcal{A}$ and the challenger.

Game 1**:** In this game, we change the way the public matrices $\overrightarrow{\mathbf{A}}$ and $\mathbf{T}_j^b$ (with $b \in \{0,1\}$ and $j \in [d]$) are generated. Upon receiving (M^*, C^*, t^*), the challenger denotes $\mathbf{y}^* = \mathsf{bitstr}(C^*, M^*)$ and does the following:

- Generates $(\mathbf{A}, \mathbf{A}_{\gamma_0}^{-1})$ as before and then samples $\overrightarrow{\mathbf{R}}_A \xleftarrow{\$} \{0,1\}^{p\ell \times m}$ and computes $\overrightarrow{\mathbf{A}} = \mathbf{A}\overrightarrow{\mathbf{R}}_A + \mathbf{y}^* \otimes \mathbf{G}$, the new distribution of $\overrightarrow{\mathbf{A}}$ is at negligible distance from the uniform distribution on $\mathbb{Z}^{n \times \ell m}$ thanks to Lemma 2 (as $p \geq 4n \log q$),
- Generates $\mathbf{R}_j^{\mathsf{Bin}(t^*)[j]} \xleftarrow{\$} \{0,1\}^{p \times p}$ and computes $\mathbf{T}_j^{\mathsf{Bin}(t^*)[j]} = \mathbf{A}\mathbf{R}_j^{\mathsf{Bin}(t^*)[j]}$, using Lemma 2, the actual distribution of $\mathbf{T}_j^{\mathsf{Bin}(t^*)[j]}$ is at negligible distance from the uniform distribution on $\mathbb{Z}_q^{n \times p}$,
- Generates $\mathbf{T}_j^{1-\mathsf{Bin}(t^*)[j]}$ for all $j \in [d]$ via $(\mathbf{T}_j^{1-\mathsf{Bin}(t^*)[j]}, \mathbf{S}_j) \leftarrow \mathsf{TrapGen}(1^n, 1^p, q)$, the new distribution of $\mathbf{T}_j^{1-\mathsf{Bin}(t^*)[j]}$ is at negligible distance from the uniform distribution thanks to Lemma 1.

  Finally, we can argue that the Game 0 and Game 1 are statistically indistinguishable.

Game 2**:** In this game, we change the way the challenger answers key queries. For a key query (i, x_i, t), there are two possibilities:

$t \leq t^*$: In this case, from the conditions that $\mathcal{A}$ has to meet in order to win the forward-secure no-signing-query selective unforgeability game after outputting the forgery, the attacker can not query a key on an attribute x_i such that $U_{x_i}(\mathbf{y}^*) = C^*(x_i) = 0$, then $U_{x_i}(\mathbf{y}^*) = 1$.
By Theorem 1, we have $(\overrightarrow{\mathbf{A}} - \mathbf{y}^* \otimes \mathbf{G}) \cdot \mathbf{H}_{x_i,\mathbf{y}^*} = \mathbf{A}_{x_i} - U_{x_i}(\mathbf{y}^*)\mathbf{G} = \mathbf{A}_{x_i} - \mathbf{G}$.
Then, we have $\mathbf{A}_{x_i} = (\overrightarrow{\mathbf{A}} - \mathbf{y}^* \otimes \mathbf{G}) \cdot \mathbf{H}_{x_i,\mathbf{y}^*} + \mathbf{G} = \mathbf{A} \cdot \overrightarrow{\mathbf{R}}_A \cdot \mathbf{H}_{x_i,\mathbf{y}^*} + \mathbf{G}$. This allows (by Lemma 6) to compute $\mathbf{R}_{x_i} = [\mathbf{A}||\mathbf{A}_{x_i}]^{-1}_{\tau_s} = [\mathbf{A}||\mathbf{A} \cdot \overrightarrow{\mathbf{R}}_A \cdot \mathbf{H}_{x_i,\mathbf{y}^*} + \mathbf{G}]^{-1}_{\tau_s}$ given $\mathbf{A}$, $\overrightarrow{\mathbf{R}}_A$ and $\mathbf{H}_{x_i,\mathbf{y}^*}$. We remark that the parameters from Lemma 6 are satisfied as $\|\mathbf{H}_{x_i,\mathbf{y}^*}\|_\infty \leq (2m)^d$ and $\sqrt{mp}\,\|\overrightarrow{\mathbf{R}}_A \mathbf{H}_{x_i,\mathbf{y}^*}\|_\infty \leq \sqrt{mp}\,\ell m \cdot \|\mathbf{H}_{x_i,\mathbf{y}^*}\|_\infty \leq \sqrt{p} \cdot \ell 2^d m^{1.5+d} \leq \tau_s$. Then, having $\mathbf{R}_{x_i}$, the challenger can compute $\mathsf{sk}_{x_i,t}$ using ABS.KeyGen and ABS.KeyUpdate algorithms.

$t > t^*$: In this case, the condition $U_{x_i}(\mathbf{y}^*) = C^*(x_i) = 1$ is not necessarily verified. Indeed, because of the forward-security, an attacker can ask a key for (C^*, M^*), for a time period $t > t^*$, and therefore $C^*(x_i) = 0$, in which case we can not compute the keys as previously. Then, for each node $z \in \text{Nodes}_{(t \to T-1)}$, the challenger first computes the smallest index $d_{z,t}$ such that $1 \leq d_{z,t} \leq d$ and $\mathsf{Bin}(t^*)[d_{z,t}] \neq z[d_{z,t}]$. Then he computes $\mathsf{sk}_{x_i}[z] = \text{RandBasis}(\text{ExtBasis}(\mathbf{S}_{d_{z,t}}, \mathbf{A}_{x_i,z}), s_{d_{z,t}})$. Finally, he sets $\mathsf{sk}_{x_i,t}$ as in the ABS scheme and sends it to the adversary.

The distribution of $\mathsf{sk}_{x_i,t}$ remains the same in both cases. Note that in the second case, when $t > t^*$, RandBasis algorithm outputs bases that are statistically close when taking as input two different bases of the same lattice. Thus, Game 1 and Game 2 are statistically indistinguishable.

Game 3**:** Finally, we change the way the challenger samples $\mathbf{A}$. Instead of sampling it with a trapdoor as in ABS.Setup, he simply samples an uniform $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, so that the distribution is close to the one in the previous game and the two games are statistically indistinguishable.
Then, we replace the challenger in Game 3 with an algorithm $\mathcal{B}$ for which we give the description below and which solves the SIS problem using a forged signature generated by the adversary.
Algorithm $\mathcal{B}$ is given $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ uniform and then he plays the forward-secure no-signing-query unforgeability game with $\mathcal{A}$ using matrix $\mathbf{A}$.
Assume that $\mathcal{A}$ produces a valid forgery $\sigma_{\mathbf{y}^*,t^*}$ for $(\mathbf{y}^* = \mathsf{bitstr}(M^*, C^*), t^*)$ given at the beginning of the game. Then $\sigma_{\mathbf{y}^*,t^*} \neq \mathbf{0}$, $\|\sigma_{\mathbf{y}^*,t^*}\|_\infty \leqslant B$ and $[\mathbf{A}\|\overrightarrow{\mathbf{A}} - \mathbf{y}^* \otimes \mathbf{G}\|\mathbf{T}_1^{\mathsf{Bin}(t^*)[1]}\| \cdots \|\mathbf{T}_d^{\mathsf{Bin}(t^*)[d]}] \cdot \sigma_{\mathbf{y}^*,t^*} = \mathbf{0}$,
and $[\mathbf{A}\|\mathbf{A}\overrightarrow{\mathbf{R}}_A\|\mathbf{A}\mathbf{R}_1^{\mathsf{Bin}(t^*)[1]}\| \cdots \|\mathbf{A}\mathbf{R}_d^{\mathsf{Bin}(t^*)[d]}] \cdot \sigma_{\mathbf{y}^*,t^*} =$
$\mathbf{A} \cdot [\mathbf{I}\|\overrightarrow{\mathbf{R}}_A\|\mathbf{R}_1^{\mathsf{Bin}(t^*)[1]}\|...\|\mathbf{R}_d^{\mathsf{Bin}(t^*)[d]}] \cdot \sigma_{\mathbf{y}^*,t^*} = \mathbf{0}$.
Since $\left\| [\mathbf{I}\|\overrightarrow{\mathbf{R}}_A\|\mathbf{R}_1^{\mathsf{Bin}(t^*)[1]}\|...\|\mathbf{R}_d^{\mathsf{Bin}(t^*)[d]}] \cdot \sigma_{\mathbf{y}^*,t^*} \right\|_\infty \leqslant (\ell(m+d)+1)\|\sigma_{\mathbf{y}^*,t^*}\|_\infty =$
$(\ell(m+d)+1)B \leq B'$,
it means that $[\mathbf{I}\|\overrightarrow{\mathbf{R}}_A\|\mathbf{R}_1^{\mathsf{Bin}(t^*)[1]}\|...\|\mathbf{R}_d^{\mathsf{Bin}(t^*)[d]}] \cdot \sigma_{\mathbf{y}^*,t^*}$ is a valid solution for $\text{SIS}_{n,q,B',m'}$. □

Lemma 9. *Our ABS satisfies FS adaptive unforgeability under the subexponential hardness of* SIS*.*

4 Forward-Secure Group Signature Scheme

In this section we present the construction of our forward-secure group signature (FS-GS) scheme from lattices.

4.1 Framework and security properties

We use the model of forward-secure group signature scheme formalized in [NHF09] and [LNWX19] and we give the definition below.

Definition 4. *A forward-secure group signature scheme consists of the following algorithms:*

$\mathsf{GS.KeyGen}(1^\lambda, 1^N, 1^T)$ *is a randomized algorithm taking as input a security parameter* λ*, number of users* N *and number of time periods* T*. Its output consists of a group public key* gpk*, an opening key* gok *and a set of initial user secret keys* $\{\mathsf{gsk}_{i,0}\}_{i\in[N]}$*.*

$\mathsf{GS.KeyUpdate}(\mathsf{gpk}, \mathsf{gsk}_{i,t}, i, t+1)$ *is a randomized algorithm that takes as input the group public key* gpk*, the secret key* $\mathsf{gsk}_{i,t}$ *of user* i *at time* t*, a user* i *and a time period* $t+1$ *and outputs* $\mathsf{gsk}_{i,t+1}$*, the secret signing key of user* i *at time* $t+1$*.*

$\mathsf{GS.Sign}(\mathsf{gpk}, \mathsf{gsk}_{i,t}, i, M, t)$ *takes as input the group public key* gpk*, the* i*th user secret key* $\mathsf{gsk}_{i,t}$ *at time* t*, the index* i *of the user, a message* $M \in \{0,1\}^*$ *and the current time interval* t *and outputs a group signature* Σ*.*

$\mathsf{GS.Verify}(\mathsf{gpk}, M, \Sigma, t)$ *takes as input the group public key* gpk*, a message* M*, a signature* Σ *and the time period* t*. It outputs either* Valid *or* $\mathsf{Invalid}$*.* Valid *indicates that* Σ *is a valid signature on* M *at time period* t *w.r.t* gpk*.*

$\mathsf{GS.Open}(\mathsf{gpk}, \mathsf{gok}, M, \Sigma, t)$ *takes as input the group public key* gpk*, the opening key* gok*, a message* M*, a signature* Σ *and time interval* t *and outputs an identity or* $\mathsf{Invalid}$ *if it fails to identify the signer.*

We require two security properties: forward-secure traceability and CCA-selfless anonymity.

Correctness. We require that for all $\lambda, N \in \mathrm{poly}(\lambda), T \in \mathbb{N}$, $(\mathsf{gpk}, \mathsf{gok}, \{\mathsf{gsk}_{i,0}\}_{i\in[N]}) \leftarrow \mathsf{GS.KeyGen}(1^\lambda, 1^N, 1^T)$, $\forall i \in [N]$, all $M \in \{0,1\}^*$, all $\mathsf{gsk}_{i,t} \leftarrow \mathsf{GS.KeyUpdate}(\mathsf{gpk}, \mathsf{gsk}_{i,t-1}, i, t)$ and for all $t \in [T]$, the following equations hold:

$$\mathsf{GS.Verify}(\mathsf{gpk}, M, \mathsf{GS.Sign}(\mathsf{gpk}, \mathsf{gsk}_{i,t}, i, M, t), t) = \mathsf{Valid}, \text{ and}$$
$$\mathsf{GS.Open}(\mathsf{gpk}, \mathsf{gok}, M, \mathsf{GS.Sign}(\mathsf{gpk}, \mathsf{gsk}_{i,t}, i, M, t), t) = i.$$

CCA-selfless anonymity. We say that a forward-secure group signature scheme provides **CCA-selfless anonymity** if no PPT adversary $\mathcal{A}$ has non-negligible advantage in the following game.

Setup: At the begining of the game, adversary $\mathcal{A}$ is given $1^\lambda, 1^N, 1^T$ as input and sends $i_0^*, i_1^* \in [N]$ to the challenger. The challenger runs $\mathsf{GS.KeyGen}(1^\lambda, 1^N, 1^T)$ to produce a public key gpk, a secret key gok and users secret keys $\mathsf{gsk} = \{\mathsf{gsk}_{i,0}\}_{i\in[N]}$ and gives $(\mathsf{gpk}, \{\mathsf{gsk}_{i,0}\}_{i\in[N]\setminus\{i_0^*,i_1^*\}})$ to $\mathcal{A}$.

Queries: At the begining of each time period, the challenger increments a counter t and notifies $\mathcal{A}$ about it. During current time interval t, $\mathcal{A}$ can make the following queries unbounded polynomially many times.

- **Signing:** On input index $b \in \{0,1\}$ and message M, the challenger generates and outputs a signature Σ generated for member i_b and period t as $\Sigma \leftarrow \mathsf{GS.Sign}(\mathsf{gpk}, \mathsf{gsk}_{i_b^*,t}, i, M, t)$
- **Opening:** When receiving a query (M, Σ, t) from $\mathcal{A}$, the challenger runs $\mathsf{GS.Open}(\mathsf{gpk}, \mathsf{gok}, M, \Sigma, t)$ and returns the result to $\mathcal{A}$.

Challenge phase: At some period $t^* \in \{1, .., T\}$, $\mathcal{A}$ chooses its target message M^*. The challenger then flips a coin $d^* \xleftarrow{\$} \{0,1\}$, computes and returns $\mathsf{GS.Sign}(\mathsf{gpk}, \mathsf{gsk}_{i_{d^*}^*,t^*}, i_{d^*}^*, M^*, t^*)$ to $\mathcal{A}$.

Queries: After the challenge phase, $\mathcal{A}$ may continue to make signing and opening queries unbounded polynomially many times. She may not make an open query for (M^*, Σ^*, t^*).

Guess: Eventually, $\mathcal{A}$ outputs d' and wins if $d' = d^*$.

We define the advantage of $\mathcal{A}$ as $|Pr[d' = d^*] - 1/2|$ where the probability is taken over the randomness of the challenger and the adversary. A forward-secure group signature scheme is said to be CCA-selfless anonymous if the advantage of any adversary $\mathcal{A}$ is negligible in the above game.

Forward-secure traceability. A group signature scheme has the **forward-secure traceability** property if no PPT adversary $\mathcal{A}$ has non-negligible advantage in the following game where he maintains a list $\mathcal{CU}$ which is set to be empty at the beginning of the game.

Setup: At the beginning of the game, the challenger runs $\mathsf{GS.KeyGen}$ and obtains $(\mathsf{gpk}, \mathsf{gok}, \{\mathsf{gsk}_{i,0}\}_{i\in[N]})$. The adversary $\mathcal{A}$ is given $(\mathsf{gpk}, \mathsf{gok})$.

Queries: During the game, $\mathcal{A}$ can make the following queries unbounded polynomially many times.

- **Signing:** On input index $i \in \{1, .., N\}$, a message M and a period time t, the challenger generates and outputs a signature Σ generated for member i and period t as $\Sigma \leftarrow \mathsf{GS.Sign}(\mathsf{gpk}, \mathsf{gsk}_{i,t}, i, M, t)$ if $(i,t) \notin \mathcal{CU}$.
- **Corrupt:** Given an index i and time moment t, the challenger returns $\mathsf{gsk}_{i,t}$ to $\mathcal{A}$ if $(i,t) \notin \mathcal{CU}$. The challenger adds (i,t) to $\mathcal{CU}$.

Forgery: $\mathcal{A}$ eventually comes up with a signature Σ^* on a message M^* and a time period t^*. We say that $\mathcal{A}$ wins the game if:

1. $\mathsf{GS.Verify}(\mathsf{gpk}, M^*, \Sigma^*, t^*) \rightarrow \mathsf{Valid}$.
2. Only one of the following two conditions is satisfied concerning the execution of the opening algorithm where $i^* = \mathsf{GS.Open}(\mathsf{gpk}, \mathsf{gok}, M^*, \Sigma^*, t^*)$:
   (a) The opening algorithm fails i.e. $i^* = \mathsf{Invalid}$
   (b) The message M^* was not sent in a signing query before and one of the following is true: $(i^*, t^*) \notin \mathcal{CU}$ or $(i^*, t^*) \in \mathcal{CU}$ but $\mathcal{A}$ did not obtain $\mathsf{gsk}_{i^*,t}$ such that $t \leq t^*$.

We define the advantage of $\mathcal{A}$ as the probability that he wins the above game, where the probability is taken over the randomness of the challenger and the adversary. A GS scheme is said to satisfy forward-secure traceability if the advantage of any adversary $\mathcal{A}$ is negligible in the above game.

4.2 Forward-secure group signature from lattices

We now describe our lattice-based FS-GS scheme which employs the FSI-ABS scheme given in the previous section and which satisfies CCA-selfless anonymity and traceability. As the ABS used is forward-secure, we show that the group signature is also forward-secure, so we consider that the lifetime of the scheme is divided into T time periods. When entering a new period of time, a new secret key is computed from the current one and afterwards the current key is deleted promptly.

GS.KeyGen$(1^\lambda, 1^N, 1^T)$ On input security parameter λ, the number of group members N and the total number of time periods $T = 2^d$, the algorithms works as follows: First sample $\mathsf{pp} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{ABS.Setup}(1^\lambda, 1^N, 1^T)$, then, for $i \in [N]$, sample $K_i \leftarrow \mathsf{SKE.Gen}(\mathsf{pp})$ and compute $sk_{x_i,0}$ as $sk_{i||K_i,0} \leftarrow \mathsf{ABS.KeyGen}(\mathsf{msk}, i, i||K_i)_{i\in[N]}$.
Output $\mathsf{gpk} = (\mathsf{pp}, \mathsf{mpk}), \mathsf{gok} = \{K_i\}_{i\in[N]}, \mathsf{gsk}_{i,0} = (i, K_i, \mathsf{sk}_{i||K_i,0})$.

GS.KeyUpdate$(\mathsf{gpk}, \mathsf{gsk}_{i,t}, i, t+1)$ It calls the key update algorithm of the ABS and returns $\mathsf{gsk}_{i,t+1} = (i, K_i, \mathsf{ABS.KeyUpdate}(\mathsf{mpk}, i, \mathsf{sk}_{t,i}, t+1))$.

GS.Sign$(\mathsf{gpk}, \mathsf{gsk}_{i,t}, i, M, t)$ In order to sign a message, the user samples $(\mathsf{ovk}, \mathsf{osk}) \leftarrow \mathsf{OTS.KeyGen}(1^\lambda)$ and computes the encryption of his identity under the key K_i as $\mathsf{ct} \leftarrow \mathsf{SKE.Enc}(K_i, i||\mathsf{ovk})$. Then, he computes

$$\sigma \leftarrow \mathsf{ABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{i||K_i}, C[\mathsf{ovk}, \mathsf{ct}], M, t),$$

where the circuit $C[\mathsf{ovk}, \mathsf{ct}]$ is defined as follows:

$C[\mathsf{ovk}, \mathsf{ct}](i||K_i)$

Hardwired constants: a verification key ovk of OTS and ciphertext ct of SKE

- Retrieve $i \in [N]$ and K_i from the input. If this is impossible, return 1.
- Compute $\mathsf{SKE.Dec}(K_i, \mathsf{ct}) = i'||\mathsf{ovk}'$. If $i' = i$ and $\mathsf{ovk}' = \mathsf{ovk}$ output 0. Otherwise, output 1.

Finally run $\tau \leftarrow \mathsf{OTS.Sign}(\mathsf{osk}, M\|\sigma)$.
The signature consists of $\Sigma = (\mathsf{ct}, \mathsf{ovk}, \sigma, \tau)$.

GS.Verify$(\mathsf{gpk}, M, \Sigma, t)$. On input gpk, a message M, a group signature Σ on M and a period time t, check that $\mathsf{ABS.Verify}(\mathsf{mpk}, C[\mathsf{ovk}, \mathsf{ct}], M, \sigma, t) = \mathsf{Valid}$ and $\mathsf{OTS.Verify}(\mathsf{ovk}, \tau, M\|\sigma) = \mathsf{Valid}$; if one of these verification condition does not hold, return $\mathsf{Invalid}$. Otherwise return Valid.

GS.Open$(\mathsf{gpk}, \mathsf{gok}, M, \Sigma, t)$. First run **GS.Verify**$(\mathsf{gpk}, M, \Sigma, t)$ and return $\mathsf{Invalid}$ if the verification result does not hold. Otherwise, parse $\Sigma \rightarrow (\mathsf{ct}, \mathsf{ovk}, \sigma, \tau)$. Since the manager does not know the identity of the user who produced the signature, he has to find it by trial and error, i.e. he computes $d_i \leftarrow \mathsf{SKE.Dec}(K_i, \mathsf{ct})$ for $i \in [N]$ and outputs the smallest index i such that $d_i \neq \mathsf{Invalid}$. If there is no such i, return $\mathsf{Invalid}$.

4.3 Security

Correctness. The correctness of the FS-GS scheme follows directly from the correctness of OTS, ABS and SKE.

We prove that a signature $\Sigma = (\mathsf{ct}, \mathsf{ovk}, \sigma, \tau) \leftarrow$ **GS.Sign**$(\mathsf{gpk}, \mathsf{gsk}_{i,t}, i, M, t)$ that was correctly generated passes the verification. We have $\mathsf{OTS.Verify}(\mathsf{ovk}, \tau, M\|\sigma) = \mathsf{Valid}$ by the correctness of OTS. Then, by the correctness of SKE we have $C[\mathsf{ovk}, \mathsf{ct}](i||K_i) = 0$ and therefore $\mathsf{ABS.Verify}(\mathsf{mpk}, C[\mathsf{ovk}, \mathsf{ct}], \sigma, M, t) = \mathsf{Valid}$.

Theorem 2 (Traceability). *If ABS is forward-secure (adaptively) unforgeable and SKE has key-robustness then the group signature scheme constructed above has the forward-secure traceability property.*

Proof. We fix an adversary $\mathcal{A}$ and we consider the traceability game that he is playing with a challenger. Let (M^*, Σ^*, t^*) be the forgery output by $\mathcal{A}$ which can be of either type:

- Type I forgeries are those for which the Open algorithm fails to identify the signer. We define E_1 to be the event that $\mathcal{A}$ wins the game and $\mathsf{GS.Open}(\mathsf{gpk}, \mathsf{gok}, M^*, \Sigma^*, t^*) = \mathsf{Invalid}$ holds.
- Type II forgeries are those for which the Open algorithm traces to an uncorrupted member or to a corrupted member that requested keys for time periods after t^*. We define E_2 to be the event that $\mathcal{A}$ wins the game and $\mathsf{GS.Open}(\mathsf{gpk}, \mathsf{gok}, M^*, \Sigma^*, t^*) = i^*$ such that $i \notin \mathcal{CU}$ or $i^* \in \mathcal{CU}$ but $\mathcal{A}$ did not query $\mathsf{gsk}_{i^*,t}$ for $t \leq t^*$.

We handle the two kind of forgeries in the following two lemmas.

Lemma 10. *If ABS is forward-secure adaptively unforgeable then* $Pr[E_1] = \mathrm{negl}(\lambda)$.

Proof. This is a proof by contradiction where we assume that E_1 happens with non-negligible probability ϵ and show how to construct an adversary $\mathcal{B}$ that breaks the adaptive unforgeability of ABS with the same probability.

The game begins when the challenger sends $\mathcal{B}$ the public key mpk. Then $\mathcal{B}$ sets $\mathsf{gpk} = (\mathsf{pp}, \mathsf{mpk})$, generates $\mathsf{gok} = \{K_i\}_{i \in [N]}$ and and gives $(\mathsf{gpk}, \mathsf{gok})$ to $\mathcal{A}$. During the game, $\mathcal{B}$ answers to the queries of $\mathcal{A}$, for a certain period of time t, as follows:

– For a corrupt query for (i,t), $\mathcal{B}$ makes a key query $(i, i\|K_i, t)$ to its challenger who returns $\mathsf{sk}_{i\|K_i,t}$. Then $\mathcal{B}$ returns $\mathsf{gsk}_{i,t} = (i, K_i, \mathsf{sk}_{i\|K_i,t})$ to $\mathcal{A}$.
– To a signing query (i, M, t), $\mathcal{B}$ answers as follows: $\mathcal{B}$ samples $(\mathsf{ovk},\mathsf{osk}) \xleftarrow{\$} \mathsf{OTS.KeyGen}(1^k)$, computes $\mathsf{ct} \xleftarrow{\$} \mathsf{SKE.Enc}(\mathsf{K}_i, i\|\ \mathsf{ovk})$ and makes a signing query $(M, C[\mathsf{ovk}, \mathsf{ct}], t)$ to its challenger who replies with σ. Then, $\mathcal{B}$ runs $\mathsf{OTS.Sign}(\mathsf{osk},\ \mathsf{M}\|\sigma)$ and returns $\Sigma = (\mathsf{ovk}, \mathsf{ct}, \sigma, \tau)$ to $\mathcal{A}$.

At the end, $\mathcal{A}$ will output a forgery $(M^*, \Sigma^* = (\mathsf{ovk}^*, \mathsf{ct}^*, \sigma^*, \tau^*), t^*)$. If $\mathsf{GS.Verify}(\mathsf{gpk},\ M^*, \Sigma^*, t^*) = \mathsf{Valid}$ and $\mathsf{GS.Open}(\mathsf{gpk},\ \mathsf{gok},\ M^*, \Sigma^*, t^*) = \mathsf{Invalid}$ does not hold then $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ outputs $(M^*, C[\mathsf{ovk}^*, \mathsf{ct}^*], \sigma^*, t^*)$ as its forgery.

We show that $\mathcal{B}$ wins the game whenever E_1 happens. We note that $\mathsf{GS.Verify}(\mathsf{gpk}, M^*, \Sigma^*, t^*) = \mathsf{Valid}$ implies $\mathsf{ABS.Verify}(\mathsf{mpk}, C[\mathsf{ovk}^*, \mathsf{ct}^*], \sigma^*, t^*) = \mathsf{Valid}$. We need to show $C[\mathsf{ovk}^*, \mathsf{ct}^*](i||K_i) = 1$ for all $i \in N$ for which $\mathcal{B}$ has made a corrupt query for time $t \leq t^*$. It is easy to see that since $\mathsf{GS.Open}(\mathsf{gpk},\ \mathsf{gok},\ M^*, \Sigma^*, t^*) = \mathsf{Invalid}$ holds, there is no $i \in [N]$ so that $\mathsf{SKE.Dec}(K_i, \mathsf{ct}^*) = \mathsf{Valid}$. We immediately have that $C[\mathsf{ovk}^*, \mathsf{ct}^*](i||K_i) = 1$ for all $i \in N$ and any time t of the corrupt query. Since $\mathcal{B}$'s simulation is perfect, we conclude that $\mathcal{B}$ wins the game with probability ϵ. This concludes the proof of the lemma.

Lemma 11. *If ABS is forward-secure adaptively unforgeable and SKE has key-robustness then* $Pr[E_2] = \mathsf{negl}(\lambda)$.

This is a proof by contradiction where we assume that E_2 happens with non-negligible probability ϵ and show how to construct an adversary $\mathcal{B}$ that breaks the forward-secure unforgeability of ABS with non-negligible probability. We denote by E the event that $\mathcal{A}$ wins the game and $\mathsf{GS.Open}(\mathsf{gpk}, \mathsf{gok}, M^*, \Sigma^*, t^*) = i^*$ such that $i \notin \mathcal{CU}$ or $i^* \in \mathcal{CU}$ but $\mathcal{A}$ did not query $\mathsf{gsk}_{i^*,t}$ for $t \leq t^*$. We consider the following sequence of games where F_i represents the probability that E occurs and the challenger does not abort in Game i.

Game 0**:** The first game is the forward-secure traceability game between adversary $\mathcal{A}$ and challenger. Assume that $Pr[F_0] = \epsilon$.

Game 1**:** In this game, the challenger makes a guess for i^* as $j^* \xleftarrow{\$} [N]$ and for time t^* as z^* at the beginning of the game and aborts if $j^* \neq i^*$ and $z^* \neq t^*$ at the end of the game. Note that the view of $\mathcal{A}$ is independent from j^* and GS.Open outputs only two possible symbols: an integer $i \in [N]$ and Invalid, therefore we have $Pr[F_1] = \epsilon/(N * T)$.

Game 2**:** Here, the challenger aborts the game when his guess (j^*, z^*) turns out to be false (meaning that $j^* \neq i^*$ or $t^* \neq z^*$) . This can happen when $\mathcal{A}$ makes a corruption query for j^* at time $t \leqslant t^*$ or i^*, t^* defined at the end of the game do not equal to j^*, z^* respectively. This change does not have any effect on the probability, therefore $Pr[F_1] = Pr[F_2]$.

Game 3**:** In this game, the challenger aborts at the end of the game if $|\{i \in [N] : \mathsf{SKE.Dec}(K_i, \mathsf{ct}^*) \neq \mathsf{Invalid}\}| \neq 1$ for the forgery $(M^*, \Sigma^* = (\mathsf{ovk}^*, \mathsf{ct}^*, \sigma^*, \tau^*), t^*)$ output by $\mathcal{A}$ at the end of the game. We show that the probability to have

E and $|\{i \in [N] : \mathsf{SKE.Dec}(K_i, \mathsf{ct}^*) \neq \mathsf{Invalid}\}| \neq 1$ happening at the same time is negligibly small. We note that E implies that $\exists i \in [N]$ such that $\mathsf{SKE.Dec}(K_i, \mathsf{ct}^*) \neq \mathsf{Invalid}$ and together with the abort condition results in $|\{i \in [N] : \mathsf{SKE.Dec}(K_i, \mathsf{ct}^*) \neq \mathsf{Invalid}\}| \geq 2$. Further we show that the probability of having the last inequality is negligible.

$$Pr[|\{i \in [N] : \mathsf{SKE.Dec}(K_i, \mathsf{ct}^*) \neq \mathsf{Valid}\}| \geq 2]$$

$$\leq Pr\left[\begin{matrix}\mathsf{pp} \leftarrow \mathsf{SKE.Setup}(1^k), \mathsf{K}_j \leftarrow \mathsf{SKE.Gen(pp)}\,\text{for}\, j \in [N] : \exists \mathsf{ct}^*, \exists i, i^* \in [N] \\ s.t.\, i \neq i^*, \mathsf{SKE.Dec}(K_i, \mathsf{ct}^*) \neq \mathsf{Valid}, \mathsf{SKE.Dec}(K_{i^*}, ct^*) \neq \mathsf{Valid}\end{matrix}\right]$$

$$\leq \sum_{i,i^* \in [N] s.t. i \neq i^*} Pr\left[\begin{matrix}\mathsf{pp} \leftarrow \mathsf{SKE.Setup}(1^k), \mathsf{K}_i, \mathsf{K}_i^* \leftarrow \mathsf{SKE.Gen(pp)} : \exists \mathsf{ct}^*, \exists i, i^* \in [N] \\ s.t. i \neq i^*, \mathsf{SKE.Dec}(K_i, \mathsf{ct}^*) \neq \mathsf{Valid}, \mathsf{SKE.Dec}(K_{i^*}, \mathsf{ct}^*) \neq \mathsf{Valid}\end{matrix}\right]$$

$$\leq N(N-1)/2 \cdot \mathrm{negl}(\lambda) = \mathrm{negl}(\lambda).$$

where the second inequality is by the union bound and the third one is by the key-robustness of SKE. Therefore $|Pr[F_2] - Pr[F_3]| = \mathrm{negl}(\lambda)$.
We then replace the challenger in Game 3 with an adversary $\mathcal{B}$ against the forward-secure unforgeability of ABS with advantage $Pr[E_3]$.
First, the challenger sends $\mathcal{B}$ the public key mpk. Then $\mathcal{B}$ sets $\mathsf{gpk} = (\mathsf{pp}, \mathsf{mpk})$, generates $\mathsf{gok} = \{K_i\}_{i \in [N]}$ and and gives $(\mathsf{gpk}, \mathsf{gok})$ to $\mathcal{A}$. During the game, $\mathcal{B}$ answers to the queries of $\mathcal{A}$ as follows for a certain period of time t.

- For a corrupt query for (i, t): If $i = j^*$ and $t = z^*$, $\mathcal{B}$ aborts. Else, he makes a key query $(i, x_i = i||K_i, t)$ to its challenger who returns $\mathsf{sk}_{x_i,t}$. Then $\mathcal{B}$ returns $\mathsf{gsk}_{i,t} = (i, K_i, \mathsf{sk}_{x_i,t})$ to $\mathcal{A}$.
- To a signing query (i, M, t), $\mathcal{B}$ answers as follows: $\mathcal{B}$ samples $(\mathsf{ovk}, \mathsf{osk}) \xleftarrow{\$} \mathsf{OTS.KeyGen}(1^k)$, computes $\mathsf{ct} \xleftarrow{\$} \mathsf{SKE.Enc}(\mathsf{K}_i, i||\ \mathsf{ovk})$ and makes a signing query $(M, C[\mathsf{ovk}, \mathsf{ct}], t)$ to its challenger who replies with σ. Then, it runs $\mathsf{OTS.Sign}(\mathsf{osk},\ \mathsf{M}\ ||\sigma)$ and returns $\Sigma = (\mathsf{ovk}, \mathsf{ct}, \sigma, \tau)$ to $\mathcal{A}$.

At the end, $\mathcal{A}$ will output a forgery $(M^*, \Sigma^* = (\mathsf{ovk}^*, \mathsf{ct}^*, \sigma^*, \tau^*), t^*)$. In this case, there are two situations where $\mathcal{B}$ aborts: if either $\mathsf{GS.Verify}(\mathsf{gpk}, M^*, \Sigma^*, t^*) = \mathsf{Valid}$ or $i^* = j^*$ does not hold where $i^* = \mathsf{GS.Open}(\mathsf{gpk}, \mathsf{gok}, M^*, \Sigma^*, t^*)$ or $t^* \neq z^*$. It also aborts if $|\{i \in [N] : \mathsf{SKE.Dec}(K_i, \mathsf{ct}^*) \neq \mathsf{Invalid}\}| \neq 1$. Otherwise, $\mathcal{B}$ outputs $(M^*, C[\mathsf{ovk}^*, \mathsf{ct}^*], \sigma^*)$ as its forgery.
In the following we show that $\mathcal{B}$ wins the game whenever event F_3 occurs. We verify that the conditions for the winner of the forward-secure unforgeability game are satisfied. Since $\mathsf{GS.Verify}(\mathsf{gpk}, M^*, \Sigma^*, t^*) = \mathsf{Valid}$ we have $\mathsf{ABS.Verify}(\mathsf{mpk},\ C[\mathsf{ovk}^*, \mathsf{ct}^*], \sigma^*, M, t^*) = \mathsf{Valid}$. We then show that $C[\mathsf{ovk}^*, \mathsf{ct}^*](i||K_i) = 1$ for any key queried by $\mathcal{B}$ respective to x_i with $i \in [N] \setminus \{i^*\}$ and t where $t \leq t^*$. We note that $C[\mathsf{ovk}^*, \mathsf{ct}^*](i||K_i) = 1$ is true since $\mathsf{SKE.Dec}(\mathsf{K}_i^*, \mathsf{ct}^*) \neq \mathsf{Invalid}$ and $|\{i \in [N] : \mathsf{SKE.Dec}(K_i, \mathsf{ct}^*) \neq \mathsf{Invalid}\}| = 1$. Moreover, $C[\mathsf{ovk}^*, \mathsf{ct}^*](i||K_i) = 1$ for any time period since the secret keys K_i involved in the circuit does not depend on the time period.
It remains to show that $\mathcal{B}$ has never made signing queries for $(M^*, C[\mathsf{ovk}^*, \mathsf{ct}^*], t^*)$. Because $\mathcal{A}$ has won the game, we have $M^* \neq M$ which implies $(M^*, C[\mathsf{ovk}^*, \mathsf{ct}^*], t^*) \neq$

$(M, C[\mathsf{ovk}, \mathsf{ct}], t)$. Therefore we have that the winning probability of $\mathcal{B}$ is exactly $Pr[F_3]$. This concludes the proof of the lemma.

This concludes the proof of the theorem since it follows that the advantage of the adversary $\mathcal{A}$ in the traceability game is negligible.

The following theorem addresses the CCA-selfless anonymity of the above GS scheme. We omit the proof and mention that it is a straightforward adaptation of the CCA-selfless anonymity proof from [KY19, Th. 5].

Theorem 3 (CCA-selfless anonymity). *If ABS is perfectly private and adaptive unforgeable, OTS is strongly unforgeable and SKE is IND-CCA secure and key-robust, then GS constructed as above is CCA-selfless anonymous.*

Acknowledgements This work is supported by the European Union PROMETHEUS project (Horizon 2020 Research and Innovation Program, grant 780701) and by the french Programme "Investissement d'Avenir" under the national project RISQ P141580-2660001 / DOS0044216.

References

ABB10. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 98–115. Springer, 2010.

ACJT00. Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270. Springer, 2000.

Ajt96. Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *STOC*, pages 99–108. ACM, 1996.

BB04. Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 443–459. Springer, 2004.

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.

BCN18. Cecilia Boschini, Jan Camenisch, and Gregory Neven. Floppy-sized group signatures from lattices. In *ACNS*, volume 10892 of *Lecture Notes in Computer Science*, pages 163–182. Springer, 2018.

BM99. Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448. Springer, 1999.

BMW03. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer, 2003.

BSSW06. Xavier Boyen, Hovav Shacham, Emily Shen, and Brent Waters. Forward-secure signatures with untrusted update. In *ACM Conference on Computer and Communications Security*, pages 191–200. ACM, 2006.

BSZ05. Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 136–153. Springer, 2005.

BW06. Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 427–444. Springer, 2006.

BY03. Mihir Bellare and Bennet S. Yee. Forward-security in private-key cryptography. In *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2003.

CHK03. Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271. Springer, 2003.

CHKP10. David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 523–552. Springer, 2010.

CL04. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer, 2004.

CNR12. Jan Camenisch, Gregory Neven, and Markus Rückert. Fully anonymous attribute tokens from lattices. In *SCN*, volume 7485 of *Lecture Notes in Computer Science*, pages 57–75. Springer, 2012.

CvH91. David Chaum and Eugène van Heyst. Group signatures. In *EUROCRYPT*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer, 1991.

dPLS18. Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler. Lattice-based group signatures and zero-knowledge proofs of automorphism stability. In *ACM Conference on Computer and Communications Security*, pages 574–591. ACM, 2018.

DvOW92. Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Des. Codes Cryptogr.*, 2(2):107–125, 1992.

GKV10. S. Dov Gordon, Jonathan Katz, and Vinod Vaikuntanathan. A group signature scheme from lattice assumptions. In *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 395–412. Springer, 2010.

Gro07. Jens Groth. Fully anonymous group signatures without random oracles. In *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 164–180. Springer, 2007.

Gün89. Christoph G. Günther. An identity-based key-exchange protocol. In *EUROCRYPT*, volume 434 of *Lecture Notes in Computer Science*, pages 29–37. Springer, 1989.

HILL99. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

IR01. Gene Itkis and Leonid Reyzin. Forward-secure signatures with optimal signing and verifying. In *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 332–354. Springer, 2001.

KY19. Shuichi Katsumata and Shota Yamada. Group signatures without NIZK: from lattices in the standard model. In *EUROCRYPT (3)*, volume 11478 of *Lecture Notes in Computer Science*, pages 312–344. Springer, 2019.

LLLS13. Fabien Laguillaumie, Adeline Langlois, Benoît Libert, and Damien Stehlé. Lattice-based group signatures with logarithmic signature size. In *ASIACRYPT (2)*, volume 8270 of *Lecture Notes in Computer Science*, pages 41–61. Springer, 2013.

LLM+16. Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In *ASIACRYPT (2)*, volume 10032 of *Lecture Notes in Computer Science*, pages 373–403, 2016.

LLNW14. Adeline Langlois, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based group signature scheme with verifier-local revocation. In *Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 345–361. Springer, 2014.

LLNW16. Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In *EUROCRYPT (2)*, volume 9666 of *Lecture Notes in Computer Science*, pages 1–31. Springer, 2016.

LMN16. Benoît Libert, Fabrice Mouhartem, and Khoa Nguyen. A lattice-based group signature scheme with message-dependent opening. In *ACNS*, volume 9696 of *Lecture Notes in Computer Science*, pages 137–155. Springer, 2016.

LNW15. San Ling, Khoa Nguyen, and Huaxiong Wang. Group signatures from lattices: Simpler, tighter, shorter, ring-based. In *Public Key Cryptography*, volume 9020 of *Lecture Notes in Computer Science*, pages 427–449. Springer, 2015.

LNWX17. San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Lattice-based group signatures: Achieving full dynamicity with ease. In *ACNS*, volume 10355 of *Lecture Notes in Computer Science*, pages 293–312. Springer, 2017.

LNWX18. San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Constant-size group signatures from lattices. In *Public Key Cryptography (2)*, volume 10770 of *Lecture Notes in Computer Science*, pages 58–88. Springer, 2018.

LNWX19. San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. Forward-secure group signatures from lattices. In *PQCrypto*, volume 11505 of *Lecture Notes in Computer Science*, pages 44–64. Springer, 2019.

LY10. Benoît Libert and Moti Yung. Dynamic fully forward-secure group signatures. In *AsiaCCS*, pages 70–81. ACM, 2010.

Moh10. Payman Mohassel. One-time signatures and chameleon hash functions. In *Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Computer Science*, pages 302–319. Springer, 2010.

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, 2012.

NHF09. Toru Nakanishi, Yuta Hira, and Nobuo Funabiki. Forward-secure group signatures from pairings. In *Pairing*, volume 5671 of *Lecture Notes in Computer Science*, pages 171–186. Springer, 2009.

NZZ15. Phong Q. Nguyen, Jiang Zhang, and Zhenfeng Zhang. Simpler efficient group signatures from lattices. In *Public Key Cryptography*, volume 9020 of *Lecture Notes in Computer Science*, pages 401–426. Springer, 2015.

PS19. Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 89–114. Springer, 2019.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93. ACM, 2005.

Son01. Dawn Xiaodong Song. Practical forward secure group signature schemes. In *ACM Conference on Computer and Communications Security*, pages 225–234. ACM, 2001.

Tsa17. Rotem Tsabary. An equivalence between attribute-based signatures and homomorphic signatures, and new constructions for both. In *TCC (2)*, volume 10678 of *Lecture Notes in Computer Science*, pages 489–518. Springer, 2017.

YLH+12. Tsz Hon Yuen, Joseph K. Liu, Xinyi Huang, Man Ho Au, Willy Susilo, and Jianying Zhou. Forward secure attribute-based signatures. In *ICICS*, volume 7618 of *Lecture Notes in Computer Science*, pages 167–177. Springer, 2012.